ARI Research Note 92-05

(1)

Individual Difference Effects in Human-Computer Interaction

Anita Kak Ambardar
Northeastern Illinois University

Office of Basic Research
Michael Kaplan, Director

October 1991

United States Army
Research Institute for the Behavioral and Social Sciences

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

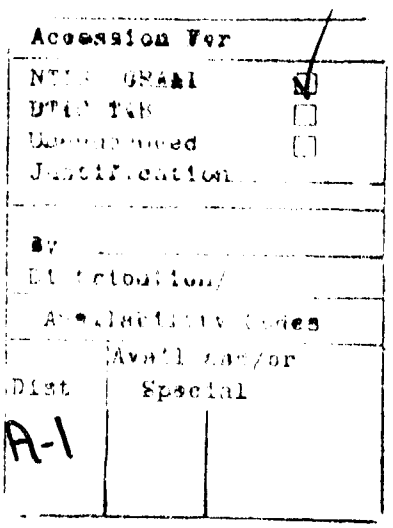
A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Technical Director

MICHAEL D. SHALER
COL, AR
Commanding

Research accomplished under contract
for the Department of the Army

Northeastern Illinois University



NOTICES

DISTRIBUTION: This report has been cleared for release to the Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or the National Technical Information Service (NTIS).

FINAL DISPOSITION: This report may be destroyed when it is no longer needed. Please do not return it to the U.S. Army Research Institute for the Behavioral and Social Sciences.

NOTE: The views, opinions, and findings in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other authorized documents.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS --
2a. SECURITY CLASSIFICATION AUTHORITY --	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE --	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ARI Research Note 92-05	5. MONITORING ORGANIZATION REPORT NUMBER(S) --
6a. NAME OF PERFORMING ORGANIZATION Northeastern Illinois University	6b. OFFICE SYMBOL (If applicable) --
6c. ADDRESS (City, State, and ZIP Code) 5500 N. St. Louis Avenue Chicago, IL 60625	7a. NAME OF MONITORING ORGANIZATION U.S. Army Research Institute 7b. ADDRESS (City, State, and ZIP Code) 5100 Eisenhower Avenue Alexandria, VA 22333-5600
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U.S. Army Research Institute for the Behavioral and Social Sciences	8b. OFFICE SYMBOL (If applicable) PERI-BR 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-82-C-0157
8c. ADDRESS (City, State, and ZIP Code) 5100 Eisenhower Avenue Alexandria, VA 22333-5600	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61102B	74F	N/A	N/A

11. TITLE (Include Security Classification)
Individual Difference Effects in Human-Computer Interaction

12. PERSONAL AUTHOR(S)
Ambardar, Anita K.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM _____ TO _____	1991, October	225

16. SUPPLEMENTARY NOTATION
Contracting Officer's Representative, Michael Kaplan

17. COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Human-computer interaction	Data base	Cognitive
			User interface design	Problem solving	style
			Human factors	Individual differences	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The optimum design for human-computer interfaces has become an increasingly important concern. Optimization requires interface to match the characteristics of the user with those of the computer system. At present, it is not clear what user characteristics, beyond a limited set related to experience and training, should be considered. This research project addresses this question.

Many forms of human-computer interaction can be viewed as problem-solving activities. Therefore, the cognitive characteristics that affect problem-solving performance may also affect problem-solving performance in the context of user interaction with a computer system.

The questions addressed in this research program were these:

1. How might fundamental, inherent individual difference dimensions of cognitive operations influence interface design?

2. What cognitive dimension, beyond experience, is really important to human-computer interaction?

(Continued)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Michael Kaplan	(703) 274-8722	PERI-BR

ARI Research Note 92-05

19. ABSTRACT (Continued)

3. Aside from the classification of users into "expert" and "novice" categories, is a given interface design equally good in supporting the problem-solving activity of any particular person?

Two experiments were conducted. The results revealed that efficiency in using computer systems depends on user cognitive characteristics and design features of computer interface. Further, usage of interface design features significantly interacts with a subject's fundamental cognitive characteristics/ cognitive style.

INDIVIDUAL DIFFERENCE EFFECTS IN HUMAN-COMPUTER INTERACTION

CONTENTS

CONTENTS (Continued)

LIST OF TABLES

CONTENTS (Continued)

INDIVIDUAL DIFFERENCE EFFECTS IN HUMAN-COMPUTER INTERACTION

SECTION 1—PART 1

Introduction

General Background of Problem Area

Perhaps the most widely accepted axiom among those concerned with the design of human/computer interfaces is that the characteristics of the interface must be matched with those of the user (e.g., Martin, 1973; Barnard, et. al., 1981).

However, a closer examination of the data bearing on the validity of this fundamental axiom reveals an interesting fact: almost without exception, the only user characteristic that has been systematically covaried with interface design is what might generally be termed "training." This body of research experience has resulted in several corollaries to the primary axiom stated above.

For example, a frequently invoked corollary for design guidance is that "novice" (untrained) users require "constrained" designs such as menu-based interface structures while "expert" (trained) users require more flexible approaches such as keyword commands.

This general concern for the importance of user training for interface design has been refined in various ways as interest in proper design of the human/computer interface has grown.

For example, Scheiderman and others have suggested that trained users be divided into several categories reflecting the type of knowledge they have derived from their past experience.

Thus, users may have knowledge of the general problem domain for which they are using the computer. Users may have knowledge concerning the syntactical and/or semantic conventions in a particular computer interface. In addition, users may possess general knowledge of computer systems and/or the specific computer systems that support their activities. The training and experience that imparts such knowledge might be termed computer-related training. It creates a dimension of distinction among users that might be termed computer-expertise.

More recently, increasing interest has focused on more general aspects of user experience and training. For example, Barnard, et. al. (1981) have considered how the syntax of a user's natural language might interact with command structure in a human/computer interface. They considered two alternative command formats: Direct-Object-First and Direct-Object-Second. These formats were thought to differ in their compatibility with the syntax of English, the native language of their subjects.

The results provided only weak support for the axiom that an interface design is improved by matching it with user characteristics. Instead, the results showed that consistency—an "experience" variable--was dominantly important in interface design. However, these studies do point out the direction of a potentially significant area of concern in the design of human/computer interfaces.

The Barnard, et. al. study is essentially concerned with the manner in which general, cognitive, rather than computer-specific, characteristics of people might interact with computer interface features. Although the cited study was basically concerned with

the impact of training, albeit training not specifically related to computers, it leads to the extended hypothesis that more basic, global cognitive individual differences among people may significantly interact with computer interface design features.

The research program described here is concerned with the manner in which enduring, basic cognitive differences among people may affect the optimum design and usability of human/computer interfaces. Thus the focus of this research program goes beyond the limited domain of training and/or knowledge-based differences among computer users to include the impact of the different cognitive styles (Witkins, et. al., 1965) and basic information processing strategies that people utilize during their interactions with computer systems.

Computer Use as Problem-Solving Activity

Very frequently people interact with computers expressly in order to solve some sort of problem. It is sometimes necessary to take a broad view of problem solving to encompass many of the forms of interaction that may be observed, but the essential features of problem-solving activity are frequently encountered.

Certainly this is true if the activity is associated with searching a computerized database. The viewpoint, expressed here, is that computer use can profitably be viewed as a problem-solving activity. That this concept may be combined with the axiom of matching computer and user characteristics has been implicitly suggested by other researchers. For example, Barnard, et. al. (1981) state that the maximum usability and effectiveness of complex (computer) systems will be achieved only if they are

compatible "...with user's cognitive skills in communication, understanding, and problem-solving" (emphasis is the present author's).

Extrinsic and Intrinsic Problem-Solving Activities—Problem-solving activity involving interaction with a computer system may be viewed at two levels:

First, the user may be viewed as attempting to obtain information necessary to solve a problem that exists beyond the domain of the human/computer interaction, per se. It is this problem that motivates the user to go to the computer terminal or other interaction site in the first place. Thus, seeking the assistance of the computer system to begin with, and actually attempting to utilize the computer at all, are problem-solving activities at the level of the external, motivating problem. This will be termed extrinsic problem-solving behavior.

Extrinsic problem-solving behavior need not involve direct interaction between a person and the computer system at all. Or, such behavior may involve interaction only through another intermediate person, usually a trained computer expert.

The second type of problem-solving activity in which computer users engage is directly concerned with the interaction that takes place between them and a computer system. The users must figure out how to "get" the needed information out of the computer or computerized database. In effect, the users must determine how to "tell" the computer* what information to supply.

*By this metaphor it is meant that the users must determine a sequence of commands and data-entry actions that will cause the computer to respond in a desired manner.

This will be termed intrinsic problem-solving behavior since it occurs within the context of a specific human/computer interface structure. The user must solve the problem of determining and then issuing a series of commands that will cause the computer to provide the help desired in the extrinsic problem-solving task.

The distinction between extrinsic and intrinsic problem-solving components is similar to the distinction made by Sackman (1977) between macro- and micro-problem solving. Sackman listed several traditional models of problem-solving and attempted to elaborate his own theory of "man-computer problem solving." Although his stated goal was to develop such a problem-solving theory at both the macro- and micro-problem (i.e., extrinsic and intrinsic) levels, his theory as presented seemed most suited for only the problem-solving steps that characterize the extrinsic level. The same can be said of the traditional problem-solving theories that he outlined.

Similar distinctions in problem-solving activity also have been made by Thomas (1978) and Innocent (1982). Thomas suggests, for example, that the problem of generating an "appropriate communication" is a separate problem level during interactive problem-solving. In the present context this would be referred to as the intrinsic level.

Typically, when the role of the computer system in problem-solving is considered (e.g., R. B. Miller, 1969) attention focuses on the manner in which the computer affects the extrinsic problem-solving process. It seems clear that this effect will depend upon the interactions among use, computer systems (including the

human/computer interface structure), and extrinsic-task characteristics.

Impact of Computers at the Extrinsic and Intrinsic Levels--The distinction between extrinsic and intrinsic problem-solving leads to another conception of the manner in which computer systems influence problem-solving behavior.

First, from the perspective of the extrinsic, motivating problem, computer systems should be designed to minimize difficulties associated with sub-problems that would normally be part of the (non-computerized) problem-solving process. Thus, if the overall process would normally involve searching professional literature for relevant citations, the computer system can offer help on this sub-problem. However, it may also occur that the introduction of a computer system does not substantially ease the difficulty of a sub-problem and, in some cases, may increase the difficulty of the extrinsic problem-solving process, perhaps by introducing new needed steps or time delays.

From the standpoint of intrinsic problem-solving activities, the computer system should be designed to minimize the difficulty of problems encountered when people use the computer system. Such difficulties are clearly related to the human/computer interface design.

Frequently, these two points of view concerning the impact of computers upon problem-solving performance are intermingled. For example, improving the quality of the command set to minimize intrinsic problem difficulty is pointless if the step being simplified is not really needed in the extrinsic problem-solving process.

One possible view is that the effectiveness of the computer at the extrinsic level is primarily task-characteristic dependent while at the intrinsic level it is primarily user-characteristic dependent.

Nevertheless, it is certainly reasonable--although, at present, there is little firm evidence to support the contention (beyond the realm of training and experience)--that the individual cognitive characteristics of computer users must be considered before the full contribution of the computer system can be achieved at either the extrinsic or intrinsic levels.

PART 2

Problem Solving and Individual Differences

Viewing human/computer interaction as problem-solving activity provides a useful perspective. An extensive literature has developed concerning the effects of differences among people along various cognitive dimensions upon problem-solving behavior. Within this context, as opposed to the context of human/computer interaction research, a broad view of individual differences has developed. Prior training, special knowledge, and experience have been but three elements of a far broader program of individual-difference research in the traditional problem-solving domain.

Within the traditional problem-solving literature, a great deal of research can be found concerning more general, fundamental, and perhaps inherent dimensions of cognitive difference among people. Among these dimensions are those grouped under the term cognitive style.

Many dimensions of cognitive style have been proposed; e.g.,

field dependence/independence, breadth of categorization, impulsive/reflective. The problem-solving literature suggests that a problem-solver's location on various dimensions of cognitive style, for example, can significantly affect problem-solving behavior. In particular, this literature reveals that the manner in which problem-solvers utilize information, the optimum format in which information should be presented, and the operational tactics and strategies devised to solve a problem, are influenced by individual differences in the cognitive characteristics of problem-solvers.

Thus, if it is valid to view human/computer interaction as typically a problem-solving process, then it is to be expected that individual differences among computer users along various cognitive style dimensions should influence the course of such interaction.

Furthermore, it is possible that characteristics of the human/computer interface design will interact with individual differences in cognitive style. Thus, the characteristics of an interface designed for a problem-solver located at one point on a particular dimension of cognitive style might not be appropriate for use by a problem-solver located at another point on the same cognitive dimension.

Unfortunately, although the literature contains many examples of how differences in background, training, and specialized expertise interact with features of a human/computer interface, they currently do not shed much light on questions of how more fundamental, inherent individual difference dimensions of cognitive operation might influence interface design. What cognitive dimensions, beyond experience, are really important to human/computer interaction? Aside from the classification of users into "expert"

and "novice" categories, is a given interface design equally good in supporting the problem-solving activity of any particular person?

Domain of Computer Use to be Considered

The principal focus of this research project, therefore, is the retrieval of information from a computerized database.

Database retrieval was selected for several reasons. First, this is a very common form of computer use. Second, the intrinsic problem-solving aspects of search tasks can be seen quite clearly. Third, it is easy to provide a realistic problem-solving task to experiment participants to maintain both the external validity of the research as well as subject motivation and interest.

Definition of Problem

There are many current definitions of "problem-solving." The primary difficulty faced in developing an adequate definition is to distinguish problem-solving from conscious thought in general. Problem-solving can be viewed so broadly that it is almost co-extensive with the term "thinking." Therefore, the term, problem-solving, potentially loses theoretical and practical utility. Problem-solving behavior exists within the context of a "problem situation." Problem-solving behavior is an activity with the goal of eliminating or reducing the problem situation in which it occurs.

The conception of a problem situation offered by Bourne et. al. (1971) involves three components:

1. A person is trying to attain some goal or to change his

present circumstances to some specifiably different situation.

2. Initial attempts have failed to accomplish this end.

3. Two or more alternative courses of action are possible.

The second condition listed above is designed to ensure that the situation is not routine, that is, the problem-solver has not previously learned some method that is immediately perceived as leading to the desired goal state. However, the present author prefers to allow that no overt failure need have occurred for a problem situation to exist; the problem-solver may have thought about and rejected many alternative problem solutions without actually attempting to carry out any of those. The critical point is that the person not perceive an immediate, routine solution method. This follows the view of problem situations offered by some other investigators (e.g., Sackman, 1981).

Thus, the working definition of a problem situation that will be adopted for this report has the following components:

1. A person is trying to attain some goal or to change present circumstances into some specifically different situation.

2. The person is not initially aware of a course of action that will lead to the desired goal.

3. Two or more alternative courses of action are perceived to exist by the problem-solver.

Implications of Problem-Solving Definition for Human/Computer Interaction--The ultimate goal of this research program is to understand and eliminate sources of difficulty in interactions between people and computers. These interactions are considered in terms of problem-solving behavior. However, since problem situations exist only at certain times (as defined in the previous

section--generally when the human is uncertain about what to do next), not all aspects of the interaction process will be relevant to this report.

For example, a human/computer interface may be difficult to use because the display is difficult to see or because a "reset" key is placed on a poor location on a keyboard and, therefore, often hit by accident. These ergonomic deficiencies do not generally result in increased uncertainty for the human and therefore, do not fall into the domain of this research program.

On the other hand, the format in which information is presented to a user may strongly affect the way that information is used and the effectiveness with which it can reduce or create problem situations. For example, a graphic presentation format may reduce problem uncertainty much more effectively for some individuals than would textual presentation of the same information.

Definition of Individual Differences

For the purposes of this report, the term individual difference is taken to signify an enduring, relatively stable characteristic of a person that has a broad influence upon their behavior and that exhibits variation from person to person.

Generally, the individual difference dimensions of concern in this report will refer to, or be correlated with, psychological/cognitive characteristics. Thus, for example, physiological characteristics such as blood pressure are not relevant individual difference dimensions.

Similarly, specific training or experience is not a relevant individual difference here because:

This characteristic can be easily changed, i.e., it is not enduring and stable.

This feature is generally specific to a limited range of situations.

The influence of some basic individual differences upon problem-solving behavior will briefly be considered here. A detailed review of this issue and dimensions of cognitive style was presented in prior reports (ref. Ambardar, 1983, 1984, Contract # MDA 903-82-c-0157).

PART 3

COGNITIVE STYLE

Background Concerning Cognitive Style

Definition of Cognitive Style—The term cognitive style has been defined as the characteristic ways in which individuals conceptually organize or structure the environment. Harvey (1963) describes cognitive style as the manner in which individuals filter and process information so their environment assumes psychological meaning. As such, cognitive style represents a mediating structure that modifies the relationship between stimulus and response. Similarly, Messick (1976) defined cognitive style in terms of consistent patterns of organizing and processing information. Zajonc (1968) has emphasized this mediating role of cognitive structure. Coop and Sigel (1971) used cognitive style "...to denote consistencies in individual modes of (cognitive) functioning in a variety of behavior situations." It should be noted that in this original definition, cognitive style is equated with behavior rather than mediating processes. (The present author has inserted the word cognitive into their phrase.)

It is clear that there are three connecting elements in all these definitions of cognitive style:

> First, cognitive style cuts across many situations to affect a broad range of behavior.
>
> Second, cognitive styles are relatively stable and enduring traits.
>
> Finally, the definitions all emphasize the structure, rather than the content of thought (e.g., Suedfeld, 1971). Structure refers to how cognition is organized; content refers to what knowledge or information is available.

Behavioral consistency is viewed as the result of this

enduring structure rather than the changing knowledge base. Therefore, characteristics of a human/computer interface matched with these structural characteristics are likely to have a longer lasting influence on performance than designs tailored to the knowledge base alone.

Historical Background on Cognitive Style—Although the first usage of the term cognitive style can be credited to Klein (1951) who described cognitive style as a "perceptual attitude," the concept of individual modes of cognition can be traced to earlier investigators.

This work primarily involved the study of personality types (e.g., Messmer, 1903; Meuman, 1907) and the most frequently used classification instrument was probably the Roschach test (Roschach, 1921). Vernon (1973) provides a thorough review of personality-based cognitive classification.

The emphasis on personality types shifted with subsequent researchers to an emphasis on perception (Klein, 1970; Smith and Klein, 1953; Gardner, 1953; Gardner et. al., 1959, 1962, 1968). Klein used the term perceptual attitude which he described as:

> "...a genotypic principle of control, with no ties to specific content and no relation to particular conflicts or stresses and having counterparts in all forms of cognitive behavior." (p. 134)

The term control was used to indicate a processing mode which was applied to incoming perceptual information. Within the context of perception, particular emphasis was given to the control processes (now, cognitive styles) referred to as leveling versus sharpening, field articulation, tolerance for ambiguous stimulus

configurations; focusing versus scanning, and field dependence/independence.

Bieri et. al. (1966) used the term cognitive structure to differentiate styles of processing perceptual information. For example, they found that people typically utilized different numbers of dimensions in classifying perceptual objects. This characteristic is currently referred to as a dimension of cognitive complexity versus simplicity.

Because of this perceptual foundation, many of the most well-known tests of cognitive style (e.g., the "Rod and Frame Test," Witkin, 1953) are clearly perceptual in nature.

However, it was gradually recognized that the concept of cognitive style could be extended to more abstract levels of information processing. Extensive evidence has accumulated that cognitive styles identified with perception are also manifested when a person deals with symbolic representations as in thinking and problem-solving. For example, individuals who experience difficulty in separating a perceived item from its environment also experience difficulty with the class or problems whose solution depends on taking a particular item out of the context in which it appears and restructuring the problem situation so that the item can be utilized in a different context.

Thus, Kagan et. al. (1963, 1964) using a object-sorting test, found that there are qualitative differences among individuals in their style of conceptualization. It was observed that subjects could be classified according to which of three common sorting rules they applied:

First, sorting by analytic-descriptive resemblance.

Second, sorting by functional association.

Third, sorting by inferential-categorical generalizations.

This trend toward a more abstract interpretation of cognitive style was also followed by Broverman (1960, 1964) who considered cognitive style to be a "relationship between abilities within individuals." The two main dimensions which Broverman developed were conceptual versus perceptual-motor and strong versus weak automatization.

The movement from a perceptual to an abstract, conceptual interpretation of cognitive style can be seen perhaps most clearly in the work of Witkin and his associated group who have developed the cognitive style dimension of field dependence/independence. Originally, this dimension referred to the relative sensitivity of an observer to the effects of surrounding context on the perception of visual objects. For example, in the "rod and frame" test, an observer is asked to judge whether a rod, surrounded by a separate square frame, is vertical or not. This judgment is found to depend upon the orientation of the frame. People are differentiated on the basis of the degree of influence of the frame on their judgment of rod-verticality.

It has been increasingly clear, however, that field independence/dependence can be interpreted more generally to reflect the relative independence of processing which an individual can achieve in dealing with related pieces of information. For example, field dependent individuals are thought to be more influenced by the local context (e.g., the preceding statements) in which an item of information is embedded, than are field

independent individuals.

It is this interpretation of cognitive style as a determinant of more complex, abstract modes of cognition that makes its study relevant to the context of human/computer interaction. For example, field dependent individuals may be more influenced by the context in which a particular menu choice appears than are field independent individuals.

Field Dependence/Independence

The cognitive style dimension of Field Dependence/Independence is probably the one which has received the most study (well over 300 studies of this dimension are reported in the literature!). The most extensive review of this dimension relative to problem-solving and learning is the one provided by Goodenough (1976). In general, this dimension has been hypothesized to affect problem-solving behavior through four mechanisms:

1. Effects on decision-making
2. Effects on information selection and utilization
3. Effects on short-term memory
4. Effects on motivation and social/situational variables

Effects on Decision-Making

There is some evidence indicative of a relationship between cognitive style and decision-making, sometimes referred to as "information management." Benbasat and Taylor (1982) in a recent review indicated that several dimensions of cognitive style are related to the decision-making process. This research has a direct bearing on the research program on information retrieval and human/computer interaction. Most of the research work conducted in

this area involving cognitive style, has been in the dimension of field independence/dependence.

Bariff and Lusk (1977) conducted an investigation into community nursing service staff. The subjects who participated in the experiment constituted the organization's decision network. The subjects were presented the same information content in four different formats. The formats varied in the complexity level, the tabular raw data being the least complex. The ogive report was the most complex. The percentage and histogram were ranked as two and three on the complexity level scale. The results of the investigation revealed that the field dependent subjects preferred the least complex data format that is tabular raw data. This finding is consistent with the low analytical and low-differentiation psychological profile which is based on Witkins theoretical framework (1974).

Benbasat and Dexter (1979) conducted a similar investigation into accounting students, professors and professional accountants. The dependent variables in the Benbasat and Dexter study were profit performance, decision time and the amount of information utilized to arrive at a particular decision.

The profit performance was evaluated in terms of the amount of sales revenue after deducting production costs. The time variable was measured in terms of the amount of time a subject spent in acquiring information and the final decision-making. The amount of information utilized in making the decision was based on the number of reports sought and the number of historical days of data asked for, per request. Forty-eight subjects participated in the

investigation. Each subject acted as an inventory/production manager of a hypothetical firm which was simulated by a computer program. The subject's task was to obtain the information and to make an appropriate decision. The results of the investigation revealed that the high analytical subjects showed higher profit than the low analytical subjects. The high analytical types took comparatively less time to make a decision than the low analytical types. The low analytical individuals asked for more structured aggregate reports than the high analytical individuals. The high analytical subjects asked for a greater number of days of historical data per request than the low analytical subjects. The results of the Benbasat and Dexter study are consistent with the findings of Huysmans 1970; Mock et. al., 1972; Benbasat and Schroeder, 1977; Vasarhelyi, 1977; Luck, 1973; Doktor and Hamilton, 1973.

In general, the findings of researchers mentioned in the preceding paragraph can broadly be premised to hold that the low analytical or field dependent individuals tend to use hypothesis testing, feedback, and trial and error approaches; whereas, the high analytical/field independent individuals tend to approach a problem in a planned manner which generally leads to the solution of the problem. Simply put, the field independent subjects take time to think the problem out before they start on the solution; whereas field dependent subjects use the trial and error method without a pre-planned approach.

Based on the consistent results of the relation between field independent/dependent dimension of cognitive style and decision-makers, Benbasat and Dexter (1982) conducted another investigation in which subjects were furnished with decision support aids. It

was based on the premise that field dependent individuals will benefit from the support aids due to their natural style of being context-dependent. Forty-eight undergraduate and graduate students participated in the study. The subjects were classified into high analytical/field independent and low analytical/ field dependent on the basis of Witkin's GEFT score. The groups were further divided into a decision aid group and a no decision aid group. The decision aid group subjects were furnished with decision aids to assist them with the decision-making process. The no decision aid group did not receive any decision aid for the solution of the problem. The results of the investigation revealed that the low analytical individual's performance improved significantly with the decision aids. The high analytical subject's performance also increased with decision aids but substantially at a lower level. The performance of the high analytical individual improved with decision aids, but there was an increase in their decision time. The decision time of the low analytical group did not increase relative to the low analytical group without decision aids. The results support the position that an individual's cognitive style plays a significant role in the decision-making process and that decision support aids do assist the low analytical individuals in decision-making. It is important, however, that the complexity of decision aids and the number of decision support aids that a low analytical individual may utilize, be consistent with his cognitive style; because if the complexity of either of the tasks is beyond what is optimum for him, the performance may deteriorate.

Effects on Information Selection and Utilization

Field Dependence/Independence was originally conceived as a dimension of distinction of information selection from a perceptual field in which the information was embedded.

In the context of problem-solving, this theoretical position has been tested primarily with three types of problem tasks: concept attainment tasks; sorting tasks; and decision-making. In all these types of tasks, subjects must selectively attend to specific features of presented stimuli to "solve" the problem task.

Concept-Attainment Problems

Davis and Frank (1979), and Davis (1975), have extensively studied performance differences of Field Dependent (FD) and Field Independent (FI) people on concept attainment problems. In these problems, people are presented with a series of multi-dimensional stimuli (e.g., stimuli may differ in dimensions of color or form). Typically, at each stimulus presentation, the subject must determine what "concept" is represented by the stimulus (e.g., the concept of "squareness") and classify the stimulus accordingly. The subject's classification is confirmed or disconfirmed by the experimenter and successive stimuli are presented until the subject achieves a criterion level of classification accuracy. Variants of this task involve Boolean combinations of stimulus dimensions to generate complex concepts (e.g., "green squares").

Davis (1975) required high school students to solve both single and compound-dimensional concept attainment problems. Subjects were classified as FI or FD on the basis of the Hidden Figures Test (French, Ekstrom, and Price, 1963). Stimulus

dimensions of color and form were used. Results of this study showed that FI subjects performed better than FD subjects on the compound-cue problems only. No significant differences were found between subject populations on the single-cue problems.

This advantage for FI subjects has been reported in a wide range of studies (e.g., Davis and Klausmeier, 1970; Dickstein, 1968; Grippin and Ohnmacht, 1972; Ohnmacht, 1966) and appears to be one of the more robust results in this area. The study of Davis and Klausmeier extended this general finding by showing that the FI advantage was maintained in "conditional" concept attainment tasks as well. In such tasks, the classification significance of a stimulus dimension depends upon the context (i.e., values of the other dimensions) in which it occurs. This situation probably has particular validity for flexible database search tasks where, for example, the meaning of a particular keyword or search command may depend upon the context in which it is given.

Davis and Klausmeier (1970) hypothesized that FI and FD subjects might differ in hypothesis testing efficiency rather than hypothesis formation ability in concept attainment problems. This hypothesis was tested in a study of 436 college students (Davis and Haueisen, 1976). Stimuli consisted of letters varying on four dimensions: color, size, letter, and position. Overall, FI subjects performed better than FD subjects (74% vs. 61% correct). The specific hypothesis under test was confirmed; FI subjects were more efficient at testing concept hypothesis although FD subjects appeared to generate an equally extensive set of potential hypothesis. Of some peripheral interest is the finding that FD and FI subjects differed in the dimensions which were preferentially used

to generate hypotheses. FI subjects preferred the color dimension while FD subjects utilized the size dimension more frequently than FI subjects.

Shapson (1977) further examined hypothesis testing as influenced by FI/FD cognitive styles in concept attainment problems. While Shapson utilized children as subjects (undoubtedly, a different population than current computer database users), his results are important because they suggest that underlying performance differences can be reduced by suitable task structuring to match the needs of particular cognitive styles.

Shapson (1977) conducted two studies with stimuli varying along four dimensions: size, color, letter (form), and position. Subjects were classified as FI or FD with the Children's Embedded Figures Test (CEFT, Karp and Konstadt, 1963). Shapson utilized a testing method which produced a more detailed picture of the information processing activities involved in hypothesis testing during the concept attainment problem-solving task. This methodology applied to the results of his first experiment revealing that FD subjects did not process information efficiently during hypothesis testing and these deficiencies were revealed in information coding, recoding, and retention. FD subjects seemed less able to focus upon single stimulus dimensions while FI subjects generally utilized an efficient "perfect focusing" strategy. (With this strategy, the subject chooses one stimulus dimension at a time, responding according to this dimension exclusively until all hypotheses related to this dimension have been disconfirmed, and only then switching to another dimension or more complex--e.g.,

compound—strategy.)

Of further interest are results of Shapson's (1977) second experiment. In this, various "aiding" manipulations were evaluated in an attempt to minimize the observed FD performance deficit. Two types of aid were found to be effective. The first involved a task aid to improve retention of information previously gathered concerning hypotheses previously entertained. These were found to be of some, but limited benefit. It was also found that the deficit could be greatly reduced by disembedding stimulus dimensions and presenting them separately. Thus, instead of presenting a large, red letter H, the three attributes were conveyed by independent cues (e.g., a red color patch, a black letter H, etc.). Shapson stated:

> Experiment 2 demonstrated that presenting stimulus materials in accordance with the cognitive style characteristics of FD children is a viable way of enhancing their information processing (1977, p. 462).

This finding may have great relevance to some aspects of human/computer interface design since it suggests that interfaces could incorporate aids that reduce problem-solving deficiencies. For example, FD subjects may do better with a sequence of several independent searches or specific commands than with a powerful compound search or a multi-function command.

Nebelkopf and Dreyer (1973) provide additional support for strong effects of FD/FI upon concept attainment problem-solving. Nebelkopf and Dreyer identified two modes of concept attainment during the problem-solving process: incremental and mediational. These modes can be identified by their characteristic learning curves. The incremental mode produces a smoothly increasing curve

(hypothesized by Nebelkopf and Dreyer to reflect classical learning processes of correct response strengths gradually building and incorrect response strengths gradually diminishing as a result of reinforcement pairings) while the mediational mode produces a disjunctive, step-function. This is hypothesized to reflect a process whereby a correct mediator (i.e., an hypothesis concerning a stimulus dimension) is selected, tested, and immediately disconfirmed or permanently retained. Since the mediational mode involves testing independent hypotheses concerning relevant stimulus dimensions, it might also be referred to as an hypothesis-testing mode.

Nebelkopf and Dreyer reported that FI subjects were generally characterized by mediational problem-solving modes while the incremental mode predominated among FD subjects. Furthermore, the problem-solving performance of FI subjects was characterized by a lack of statistical dependence across successive trials (i.e., when a stimulus dimension was disconfirmed, another was selected independently).

Nebelkopf and Dreyer suggested that FD people are not able to pick out the significant dimensions of a concept, or problem, due to their "global" style (i.e., inability to separate a stimulus dimension from the context in which it is embedded), and therefore are unable to form an effective hypothesis concerning individual dimensions. Nebelkopf and Dreyer also suggested that, while FD individuals might be deficient in mediational mode problem-solving, FI people should be able to use either incremental or mediational modes, however, FI people choose hypothesis testing (i.e., the mediational mode) since it is consistent with their previous experience.

Goodenough (1976) presents data bearing on two general features of the FD/FI distinction. The first feature distinguishes between "participant" and "observer" roles in problem-solving and learning. Goodenough describes these modes as follows:

> One approach has been variously described as passive, abstractive, or iconic. The learner has a spectator role. In contrast, learning may proceed by a process described as active, mediated, or hypothesis testing, with the learner having more of a participant role (1976, pp. 678-679).

Goodenough (1976) proposed that this distinction underlies many of the performance differences observed in concept attainment problem-solving between FD and FI subjects. For example, Nebelkopf and Dreyer's (1973) findings of different learning curves (continuous vs. discontinuous) for FD vs. FI subjects are claimed to reflect this underlying active/passive FI/FD distinction.

This distinction is also claimed to be significant beyond concept attainment problem-solving. For example, in paired associate learning tasks, FI subjects are claimed to be more active organizers of presented verbal materials and more likely to develop memory-aiding idiosyncratic mnemonic schemes (Rosencrans, 1955; Witkin et. al., 1962, 1974). This effect may have implications for human/computer interface design as well, particularly in the way commands are abbreviated, for example.

The second general feature of the FI/FD dimension concerns the role of cue salience. Goodenough (1976) points out that FD people are generally found to perform less well on concept attainment problems than FI subjects and this is particularly true of multi-dimensional problems. Goodenough points out that FD people may be particularly susceptible to cue salience. A highly salient cue dimension may capture the attention of FD problem-solvers to the

exclusion of other relevant, but less salient dimensions. Cue dimensions may be salient either because of inherent perceptual factors (e.g., color may be generally more salient than form) or because of past experience. Thus, if a problem-solver has solved a series of problems using this stimulus dimension of size, poor performance may occur when another dimension, say color, becomes relevant for solution of subsequent problems. In support of this hypothesis, it has been observed that FD people do less well on reversal-shift sequences of concept attainment problems than do FI individuals (Massari and Mansfield, 1975; Ohnmacht, 1966; Zawel, 1969).

While there are several apparent conflicts among the various studies in this area, there is good general agreement as well. Regarding conflict, for example, Nebelkopf and Dreyer's (1973) studies suggest that FD subjects are deficient in abstracting stimulus dimensions for testing (presumably hypothesis formation) while other researchers have suggested that the deficiency arises in hypothesis testing--presumably a process that occurs after an hypothesis has been formed.

Regarding agreement, several points are evident. First, FI/FD consistently has been found to influence performance in concept attainment problem-solving. These effects are strongest for multi-dimensional stimuli. It is also apparent that some of the apparent FD deficit in dealing with multi-dimensional stimuli can be reduced by separating or disembedding the various dimensions. Finally, it appears that such disembedding would not be harmful to FI subjects. However, some caution is warranted. Nebelkopf and Dreyer (1973)

for example, did find preferential learning strategies among FI subjects.

It is interesting to conjecture how the effect of disembedding would interact with Garner's (1965) stimulus dimension of integrity/separability. Presumably, separable dimensions would benefit less from the disembedding manipulation than would integral dimensions.

Another question that remains after review of existing literature is the extent to which these effects would appear in the context of non-visual (e.g., semantic) tasks. The overspanning conceptualization of FI/FD would suggest that these effects would be found in more abstract domains as well. These are the domains that would be particularly relevant to problem-solving interactions with a computerized database.

Differential Feedback Effects

One additional finding concerning FD/FI differences in concept attainment problems concerns the effects of informational feedback. A number of studies have reported that performance deficits for FD, as compared to FI problem-solvers can be traced to trials on which "wrong" feedback (i.e., the problem-solver is told that his response is wrong) is given. Positive feedback, although often less informative, seems more effective with FD problem-solvers.

Sorting Tasks

The second important problem solving domain in which FD/FI effects on information selection and utilization have been studied is sorting. In the sorting paradigm, subjects are given a group of multi-dimensional stimuli (either sequentially or all at once) and

are asked to sort the stimuli into smaller categories.

Although sorting tasks are not generally characterized as having a "right" or "wrong" answer, the categorizations which subjects produce reflect upon two primary issues: first, what dimensions do subjects use to form categories; and second, how diverse are the stimuli within a category (i.e., the extent of stimulus generalization).

Kieron, O'Connor and Blowers (1980) provide a representative study which also points to an underlying information processing difference between FD and FI people with possible relevance to human/computer interaction. In this study, 20 subjects (mean age = 29) were classified as FI or FD on the basis of the Rod and Frame Test (RFT). Each subject then performed a Color/Form Sorting Task (CFT). In this task, a series of pictures differing in both form and color dimensions was presented. Subjects classified each picture as belonging to one of two categories (each represented by a prototypical picture). In this study, it was found that FI and FD subjects consistently used different dimensions for performing this classification. FI subjects preferred to use the form dimension while FD subjects utilized the color dimension. This morphophilia-chromophilia distinction has been observed in other sorting tasks as well as concept attainment tasks.

While interesting in itself, the more general interpretation of this finding is of greater potential importance to problem-solving. Broadbent (1970) has proposed a distinction among methods of information selection (more specifically, information reduction). Filtering referred to the selection of input (i.e.,

perceptual) information on the basis of consistent physical dimensions such as color. Response selection, in contrast referred to information reduction by limiting the number of output (i.e., response) categories used by a subject.

Although one might argue with Kieron et. al.'s (1980) contention that use of the form dimension represents a limitation of response categories, there does not seem to be a consistent trend in this literature; FD subjects prefer to use filtering (and focus upon particular stimulus dimensions accordingly) while FI subjects tend to use response selection. This effect has also been reported in the concept attainment literature as described previously.

This difference could have an impact on the manner in which FI and FD subjects interact with a computerized database. Two potential impact areas are command syntax (or position-consistent forms or menus, for example), and the apparent database structure. It might be hypothesized that apparent structures, which can be visualized in rigid structures related to physical dimensions, might be best for FD subjects. An example of such an apparent structure might be a linear structure with each record in the database being sequentially numbered. Thus, the FD database searcher could specify retrieval of records 90-105 for example. A different apparent structure would be one based, for example, on keywords. Here the search dimensions (the set of acceptable keywords) does not lead to a simple mental model compatible with the stimulus filtering preference of FD subjects.

Concerning stimulus generalization, sorting tasks typically have shown a greater degree of stimulus generalization for FD people when simple physical dimensions (e.g., color) have been used

(Goodenough, 1976). This finding also has a counterpart in conditioning studies which also reveal steeper generalization gradients among FI individuals (again, primarily for simple physical continua such as brightness) (Van-Veen, et. al., 1973).

Effects on Memory

The effects of Field Dependence/Independence upon problem-solving via memory have generally been investigated in the context of learning tasks. Although this includes studies of improvement on successive concept attainment problems, paired associate learning tasks and other non-problems, paired associate learning tasks and other non-problem solving paradigms have also been used and must be considered. Clearly, differential learning processes could influence the interaction between a person and a computer system during a problem solving scenario, and this effect would be magnified for inexperienced users.

Data suggest that differences in memory "efficiency" distinguish FI and FD subjects in concept attainment problem-solving. In particular, it has been suggested that FD subjects may be less able to recall hypotheses which have been tested and this leads to the FI/FD performance differences described in previous sections (e.g., Eimas, 1970). Shapson (1977) tested the hypothesis that this type of memory difference accounted for observed FD/FI concept attainment performance differences. In his study, the effect of a memory aid which placed + and - signs over previously presented stimuli was investigated. Although the effect of memory aid alone did not reach statistical significance, this aid did

prove effective in combination with other information processing aids.

In addition, other studies (e.g., Davis and Klausmeier, 1980) found that the difference between FI and FD individuals in concept attainment tasks increased as the number of potential hypotheses grew. This memory hypothesis seems to be a memory-based analog of the perceptual-based distinction upon which the dimension of Field Dependence/Independence is based: Just as FD individuals are claimed to have more difficulty in distinguishing one stimulus dimension from among an embedding context of others in a presented complex stimulus, so are FD individuals hypothesized to have more difficulty in identifying selected hypotheses (i.e., those which have been previously tested) from the set of all possible hypotheses.

Outside the context of concept attainment problems there is some additional evidence for FI/FD differences in memory. However, the majority of studies utilizing traditional free-recall paradigms have failed to reveal consistent differences (e.g., Beischel, 1973).

To the extent that differences in free-recall memory tasks have been found, they are often claimed to be related to differences in category clusterir For example, Meshorer (1969) required subjects to categorize items in a word list prior to learning them. A measure of output clustering in the recall of the memorized words revealed more clustering for FI subjects than for FD subjects. However, despite these differences in clustering, actual recall performance was similar for FI and FD subjects.

An interesting study by Fleming (1968) has potential application to human/computer interactions. Fleming examined free-recall of word lists containing hierarchically organized items. Two conditions were compared. In one condition, the sequence of to-be-memorized words began with the superordinant category names, followed by subordinant category members. In the second condition, this order was reversed. It was hypothesized that FD people are less likely to supply their own organizations to to-be-remembered items and thus would show improved performance with the superordinant-first lists.

This result was confirmed: FI and FD subjects performed equally well when organizing names occurred at the start of the word lists. However, FI performance was superior in the reverse condition. This finding may have importance in such areas as menu organization or keyword choices in human/computer interface design. In general, the above cited results suggest that FI individuals take a more active role in restructuring presented information and this sometimes leads to improved recall.

Another area which supports the claim that FI individual's tendency toward more active structuring affects retention and learning, is that of interference studies in associative learning. Since it is thought that active organization of to-be-remembered information leads to improved encoding and immunity from memory interference, FI and FD individuals should show differential effects in interference paradigms.

Gollin and Baron (1954) suggested that the organizing ability of FI individuals would help them keep original and interpolated word lists distinct in a retroactive interference paradigm.

Although they found no difference between FI and FD subjects on learning of the original list, FI subjects were found to be significantly better in subsequent relearning of the original list.

One of the most direct tests of the influence of FI/FD upon memory was conducted by Berger and Goldberger (1979). In this study, a variety of short-term memory tasks were administered to FI and FD subject groups. In particular, the memory tasks differed in the degree to which they involved interference (e.g., long digit span) or a factor termed "registration." These factor loaded tests involved very short retention intervals and proactive inhibition. (Tests which loaded the interference factor primarily involved retroactive interference.) Results of this study showed that FI/FD differences appeared only on tasks involving retroactive interference with FI subjects producing better performance. Berger and Goldberger attributed this to the active organizing processes which FI subjects were more likely to apply to presented, to-be-remembered items.

Overall, the literature concerning the effects of Field Dependence/Independence upon memory (and consequently upon problem-solving and learning) is that these effects are fairly weak. However, they also suggest that they occur because FI subjects are more effective organizers of to-be-remembered information. Furthermore, to the extent that these effects occur, the FD detriment can sometimes be removed by providing appropriate memory aids.

Effects on Motivational and Social/Situational Variables

It has been hypothesized that FD vs. FI performance differences in a variety of tasks, including problem-solving, may

be due to differences in motivation stemming from social/situational factors. In particular FI and FD individuals appear to differ in their attention to and processing of social cues and rewards such as facial expressions or verbal reinforcement or punishment.

The most dramatic context for these effects is in incidental learning tasks. In these paradigms FD subjects are sometimes (though not with complete consistency) found to do better than FI subjects (e.g., Messick and Damarin, 1964; Eagle, et. al., 1969). This finding, which is restricted to incidental learning of socially significant items (including words), is in contrast to other learning and memory studies which, as indicated above, generally show superior performance from FI subjects.

It should be noted that this FD advantage occurs only under special circumstances. Eagle et. al., (1966) contrasted memories of FD and FI subjects for incidentally-presented words under several experimental conditions. The incidentally-presented words could be either socially relevant or not, and either task relevant or not. In this study, FI subjects produced better incidental memory performance in all conditions except that involving task-irrelevant socially relevant words; in this condition there was no difference between FI and FD groups. Thus, the task-relevancy as well as the social content of incidental material may be important in producing the FD advantages mentioned previously.

Related to this effect of socially significant information is a pattern of studies showing differences in the effectiveness or various types of intrinsic reinforcement and feedback in learning

and problem solving. These studies can be summarized as follows: in the absence of external reward or punishment, FI individuals are generally superior in learning and problem solving tasks. Second, FD individuals are more strongly affected by negative social reinforcement than are FI individuals.

Bell (1964) performed a concept learning task in which subjects could choose task versions leading to either punishment avoidance or reward (i.e., negative or positive reinforcement). Results of this study indicated that FD subjects preferred the punishment avoidance condition while FI individuals preferred the other condition. Similarly, Konstadt and Forman (1965) found that FD performance in a letter cancellation task declined more than FI performance in the presence of verbal criticism.

This sensitivity to verbal punishment has sometimes been found to produce superior performance in FD as compared to FI subjects. Ferrell (1971) found that FD subjects learned an avoidance task more rapidly than FI subjects in conditions which included verbal punishment.

Finally, it has been found that performance of FD individuals declined more rapidly than did that of FI individuals when external rewards were removed from a task (e.g., Fitz, 1979; Gates, 1971). In general, these differential effects of reinforcement may have some significance for human/computer interface design. An example might be in the area of error messages.

Summary of Field Dependence/Independence Effects Overall

It is clear that this dimension has consistent effects upon problem solving performance. Although FI individuals are most

commonly found to produce performance superior to that of FD individuals, these differences can be minimized by providing proper task aids (e.g., memory aids, cue disembedding, etc.) or task structure. Also, FD individuals may show superior performance when socially relevant stimuli are used as informational feedback (since FD individuals attend to such stimuli to a greater degree than FI individuals). Finally, punishment avoidance is an effective motivation for FD individuals while reward attainment is most effective for FI individuals.

The review of literature relating individual cognitive style differences to problem-solving provided an intuitive rationale for the notion that cognitive style dimension might be potent in human/computer interactions. Such intuitions seem, for example, to underlie the selection of ten cognitive style variables by Andriole (1982) as worth studies in the context of computerized decision-aiding systems. Additionally, the results of the literature revealed that the vast preponderance of concrete information concerning effects of cognitive style upon traditional problem-solving behavior is restricted to three dimensions of cognitive style; Field Independent and Field Dependent, impulsive/reflective and breadth of categorization. However, maximum research effort has been devoted to the Field Independent/Field Dependent dimension. For this and other reasons FI and FD dimension is the focus of this research program. The other reasons included: a) the strongest documented track record in terms of reliable effects of problem-solving performance; b) theoretical foundation and; c) highly reliable and well tested standardized assessment and evaluation instruments.

SECTION 2

OBJECTIVES OF THE RESEARCH PROGRAM

1. How do fundamental, cognitive individual differences influence interface design?

2. What cognitive dimension, beyond experience, are really important to human/computer interaction?

3. Aside from the classification of user's into "expert" and "novice" categories, is a given interface design equally good in supporting the problem-solving activity of any particular person?

The primary goal of the present research is to establish the extent to which more global, enduring, and basic cognitive characteristics of the target user's must be considered in the design of human/computer interfaces. Specifically, the hypothesis being tested is that individual differences in various dimensions of user's cognitive styles interact with features of human/computer interfaces. Furthermore, by taking these interactions into account such interfaces can be optimized to user populations of differing cognitive styles.

The project of this nature is the first attempt; therefore, it was feasible to first check the validity of the hypothesis by conducting an experiment to evaluate the information retrieval performance of subject groups differing strongly in Field Independent/Field Dependent dimension.

EXPERIMENT I

The goal of Experiment I was to quickly assess whether a sensitive design could detect effects of individual cognitive style differences upon a complex database utilization task. Essentially, the question asked was: Do people with different cognitive styles perform in characteristically distinct ways when interacting with a database retrieval system? If the answer was "yes" the next step would be to target virtual interface studies to explore the nature of these effects. If "no" was the answer, the source of the failure could be sought before investing a large amount of resources in an ineffective manner.

Finally, Experiment I provided a test vehicle to ensure that all data collection, transfer, and analysis systems were operational.

METHODS OF PROCEDURE

Two subject groups differing on the cognitive style dimension of field dependence/independence were contrasted. The subjects in each group were at comparable, novice levels of computer experience.

Subjects

Subjects were recruited through advertisements placed around the UNI campus, through newspaper ads, and through classroom solicitations. Subjects were first pre-tested to assess their locations on the primary cognitive style dimension, field dependence/independence.

Assessment of field dependence/independence was done with a

standardized test, the Embedded Figures Test. This is a paper and pencil test with reported test-retest validity coefficients ranging from .89 - .92 and corrected odd-even coefficients ranging from .90 - .95. Subjects were pre-tested individually.

If qualified on the basis of their scores, subjects were contacted by phone and an appointment for the second experimental session was made.

The entire testing session took approximately 1/2 hour. Since it was desired to have a relatively homogeneous group with respect to computer experience, only subjects who described themselves as having little or no specific computer training or expertise were used.

Equipment—Hardware

The primary computer system consists of two Apple II+ microcomputers linked by a 9.6 KB RS-232 communication line. One computer is referred to as the E-computer (experimenter's computer) and serves as the experimenter's control station during experimental sessions. From the E-computer, the experimenter can interpret responses made by the subject on the S-computer (subject's computer), the second computer in the system. The experimenter, after interpreting the subject's response, can interrogate remote or local databases from the E-computer in real time and then cause a display to appear on the S-computer in accordance with the rules of the virtual interface in effect.

In addition to this basic tandem computer system, an additional disk controller was utilized so that one of the two available system disk drives can be used on the S-computer. Also,

a high-resolution text display card has been added to the S-computer along with an additional 16K of dynamic RAM memory.

Software

Experiment Program CHAMP.4

An existing personal finance program was extensively modified for experimental use. The program provides the user five basic functions, accessible from a main menu.

The functions are:

Adding:

This allows the user to add new records (checks, deposits, and service charges) to the database.

Viewing:

This allows the user to view any particular record. Two record-access methods are provided, numerical index and keyword search. The first method allows the user to think of the database as a linear file of records. The second method allows the user to structure the database in more flexible, multidimensional ways. Keyword searches on 8 fields (e.g., amount, payee, data, special notes, type of payment, etc.) are permitted. This search flexibility is crucial. It was hypothesized that FI users might use different search strategies than FD users.

Changing:

This function allowed modification of records. It was similar in some respects to the function in that subjects had to search for and retrieve a record which they wished to change.

Balancing:

This function provided a summary balance sheet whenever requested. Thus, the subject could keep track of balance outstanding, for example, each time a change was made to a record.

Posting:

This function allowed the user to change the status of a record from "Outstanding" to "Posted." Note that this function was automated but the same result could be achieved by using the function to modify the field of a record. It was hypothesized that some users might prefer and use a small

number of powerful commands while others might prefer a larger set of more specific, targeted commands.

The details pertinent to the operation of the program and other details have been discussed in Part I, Part II (Ambardar, 1983).

In addition to these basic functions, CHAMP.4 maintained a thorough data profile. Times of every display presentation and every subject response were recorded to the nearest second. In addition, the contents of every subject response and of every display viewed by subjects were recorded. The data collection available in CHAMP.4 is complete enough to allow total reconstruction of the experimental session if that is required.

Experimental Tasks and Procedures

Subjects were tested individually in single, three-phase sessions. In Phase I the task was to learn to use CHAMP.4. In Phase II subjects created a database. In Phase III subjects were given a bank statement and told to reconcile their own database to the bank statement, locating and correcting any errors in their own database. Details of the three phases are as follows:

In Phase I, a structured learning program was provided for subjects. After being given some general familiarization with the computer and task, subjects were asked to carefully read and follow the examples in an instruction booklet. As they followed the exercises in the instructions, subjects used all the functions available in the system and saw all possible system messages at least once. They were told that they could ask questions to the experimenter (who was present during the sessions). Subjects

averaged about 1.5 hours to learn the system. There were three criteria to indicate the end of the learning phase. First, after properly completing all the instruction steps correctly, a small database would have been created and this database would have a specific, known balance. Subjects had to satisfy the criteria of attaining this final balance. Second, subjects were told to tell the experimenter when they felt they were "fairly confident" about how to use the system and that they felt they had learned the system sufficiently. Finally, since the experimenter was present during the learning phase, the experimenter used his judgment to continue the learning process if there were indications of misunderstanding or uncertainty.

In Phase II, subjects were given a printed list of 50 items which were to be added to the small database which resulted from Phase I. Phase II had two specific purposes. First, it provided subjects a chance to use the Add function and continue to learn about the system. Second, and more importantly, it gave subjects familiarity with the financial system which would form the basis of Phase III. During Phase II, subjects entered checks payable to the same payees who would be present in the Phase III consisted of one month's payments, deposits and service charges.

Thus, when subjects were given the bank statement to reconcile in Phase III, they had fairly clear knowledge of to whom checks were made (therefore, they could use these names in keyword searches, for example), their monthly salaries, service charges, etc.

In Phase III, subjects were given a bank statement which listed, in the order or posting at the bank, the amounts of all

deposits, checks and service charges made against the account. Starting and ending balances were provided. Check numbers were NOT included on the statement. Subjects were explicitly told that the bank statement was correct. They were to determine whether their own database balance agreed with the bank statement. If it did not (and it never did!) they were to utilize the CHAMP.4 system to locate and correct all errors in their database and to achieve a reconcilement of the two balances.

To do this subjects had to change the status of unposted (in their own database) checks and deposits to reflect their appearance on the bank statement. They also had to add service charges which appeared on the bank statement. Finally, the database which was provided to subjects had a large variety of errors, designed to make the task difficult. This set of errors included posting errors, amount errors in both checks and deposits, spurious service charges, and incorrectly spelled or abbreviated payee names. The set of errors were the same for all subjects.

The program recorded a complete picture of everything that occurred during Phase II and Phase III. Unlike Phase I, the experimenter was not in the same area as the subject but monitored the proceedings from an isolated, but nearby, station.

RESULTS

Training Phase: Overly structured, step-by-step instructions were used to guide subjects through the CHAMP training phase. Subjects were required to read the directions and enter the correct keys as and when instructed. Since the training phase was highly structured, one may expect similar performance from all subjects.

However, this was not the case. Significant differences in time and accuracy, as a function of cognitive style, were evident.

Accuracy for the training phase was measured by the number of keystrokes deviating from the instructions. FI subjects used significantly less keystrokes than FD subjects in all five CHAMP functions (Add, View, Change, Balance, Post). The only place for individual choice of key selection was in finding a check in changing mode, using either keyword (K-W) or item number search strategies. Field Dependent subjects were split concerning search patterns with 8 subjects using K-W search, 7 using item number search and 2 using both search methods to find the check. However, FI subjects showed a preference for K-W search, 6 (28%) used item number search and 1 subject used both methods to recall the check. Related to this, when FD subjects did use K-W search to recall the check, they made significantly greater errors by recalling the wrong check (False Alarms). These results, although limited to a single search, suggest definite search preferences depending on an individual's cognitive style.

Along with using less keystrokes, FI subjects used significantly less time during the training phase. Field Dependent subjects took an average of 4039 seconds to complete this phase while FI subjects completed training in 2570 seconds, $t(36)=3.53$, $p=.001$: 36.4% faster than FD subjects. This speed difference was distributed throughout all sections of the training phase.

These results indicate that speed and accuracy differences exist as a function of cognitive style (FI/FD). Field Independent subjects are faster, more accurate and prefer K-W search mode,

while FD subjects are slower and less accurate with no obvious search preference in the training phase.

The most striking finding thus far is a strong difference in the use of keyword vs. numerical index search methods by FI and FD subjects. FI subjects utilized keyword searches 54.2 (mean) times in Phase III while FD subjects used this mode only 17.1 times, $F(1, 20)=10.23$, $p=.005$. Conversely, FI subjects used numerical index searches only 13.0 times while FD subjects used this mode 44.7 times, $F(1, 20)=4.12$, $p=.05$. This effect can also be seen by comparing percentages of use of each search mode by the two groups: The FI groups used the keyword mode 33.2% of the time, $F(1, 20)=4.12$ $p<.05$. Clearly, keyword searching was much preferred by FI subjects than to FD subjects, while the converse was true for the linear, numerical index search mode.

SUMMARY AND CONCLUSIONS

These data are tentative, but they provided some of the first direct evidence that basic, underlying psychological characteristics can influence the nature of the human/computer interaction and that interface designs can be tailored to accommodate these specific preferences.

The results, though tentative, revealed a strong differential preference for search mode type between field independent and field dependent subjects. Field dependent subjects strongly preferred the strictly organized sequential item number mode, and field independent subjects preferred key-word search mode.

Another difference observed was in command function usage. Field independent users used view for examining and change for

altering items. Field dependent users used change for both changing and examining.

The results support the basic hypothesis that interactions affecting the efficiency with which people use computer systems arise between fundamental user cognitive characteristic and design features of human-computer interface.

The results raised several questions. 1) Why did field independent and field dependent users prefer different interface features? There are several possibilities for this user preference. In key-word mode the user is presented with relatively unstructured database view. This structure or organization which users see may be that which they themselves impose. What this structure might be is unclear, but the important point is that it is generated by them and may be complex. This complex database structure might be comfortable for field independent subject users who are able to identify search paths to information embedded in a complex structure.

In contrast, the item number search mode, preferred by FD subjects, presents highly structured straightforward database view--that is--a simple linear ordering. Perhaps the preference exhibited by FD users reflects their preference for less complex database views.

Another question raised is in difference between FD and FI subjects in functional command usage--"SEARCH VIEW" and "CHANGE COMMAND."

Perhaps preference among FD subjects to use the change command for both examining and altering items reflects general preference for interface command systems based on a small number of general

purpose commands. In contrast, FI subjects who used "VIEW" for examining and "CHANGE" for modifying a target prefer command system containing a large number of more specialized commands.

These and other questions were raised in Experiment II.

SECTION 3

EXPERIMENT II

Objectives

To directly test and validate the major implications of Experiment I—that is, cognitive characteristic differentiating FD and FI users significantly interact with interface design features.

Furthermore, results of Experiment I suggest that users have preferred modes of database search, they do not indicate whether performance of FD users would be worse if only the Keyword mode was available or if field independent subjects would perform worse if only numerical Index Search was available to them.

Specific Hypotheses

1. Obtaining information from computerized databases is a problem solving task with varying levels of difficulty. Current human problem-solving data indicate the importance of cognitive individual difference dimensions. Since database searching is to be viewed as a problem-solving activity, individual difference dimensions also will significantly affect this problem subclass.

2. Human-computer interface design can be optimized to individual differences in cognitive style among system operators.

3. Properly matching interface design to individual characteristic will significantly improve the task performance of the human operator.

METHODS OF PROCEDURE

Subjects

Subjects were recruited through advertisements placed around the university campus, through newspapers and classroom solicitations. Subjects were first pre-tested to assess their location on the primary cognitive style dimension, Field Independence and Field Dependence. Subjects scoring one sigma above and below the neutral point participated in the experiment. Assessment of FD/FI was done with a standardized test, the Embedded Figures Test. One hundred and forty-four subjects, seventy-two classified as FI and seventy-two classified as FD served as subjects. All the subjects were computer naive, that is, they had little or no computer background. The subjects were paid five dollars per session for their participation.

Equipment

Hardware/Software Configuration:

The experiment was conducted using an experimental system known as the Virtual Interface System (Appendix A). This system, which was developed as part of this project, allows rapid simulation of many types of user-interfaces within the context of a database retrieval application.

This system utilizes two Apple II+ computers communicating via a 300 bps RS-232 link. The Experimenter controls the experiment from one computer--E-computer, while the subject interacts with the system at the S-computer.

The E-computer is equipped with two disk drives for data storage as well as clocks and port boards to support the

intercomputer communication. The S-computer is equipped with one disk drive for program storage, communication ports, and a VDU.

Also, at the E-computer station, a second VDU is provided which is bridged to the subject's VDU. This allows the experimenter to monitor everything that appears on the subject's display.

Basically, the system operates as follows: A set of rules is generated to define the command syntax for a particular interface. Display formats, consistent with these rules, are created using Virtual Interface System's frame-creation features. The subject is given a problem requiring database search and enters commands at the S-computer keyboard. These commands are passed to the E-computer where they are interpreted by the experimenter according to the rules of the interface currently in effect. The experimenter then generates a response, enters it at the E-computer keyboard, and causes the response to be sent back to the S-computer where it appears on the subject's VDU.

The E-computer automatically records a large quantity of data during system operation. This includes the time (to nearest second accuracy) and content of all inter-computer transactions as well as some local activities at the S-computer (reviewing of problems, help-requests). Thus the entire experimental session could be effectively recreated if desired.

All information (aside from the static help screen and "problem" screen) were presented within an 80-column x 23-row display format using inverse-video.

Apart from the computer equipment, the testing room is configured with a partition between the subject and experimenter areas so that the subject does not know that his/her responses are

actually being processed and observed by the experimenter. However, the experimenter can overhear spoken comments made by the subject.

Interface Structure

Twelve experimental interfaces were [illegible]. (See Table 1)

(The exact specifications of each interface are contained in Appendix B.) The following descriptions of interface features are designed to give a general idea of the nature of each of the twelve interfaces.

The twelve interfaces represented three main types:

1. Sequential--The subject interacted with the database as though all records were simply listed in a sequential, alphabetically-organized list. Thus, the subject might ask to see items 20-25. This interface type was designed to impose the greatest degree of constraint and simplicity upon the user's search strategies and upon the user's view of the database structure.

2. Keyword--The subject used a keyword-based command language to retrieve items from the database. The set of acceptable keywords was not given to the subject. The subject had to generate a search strategy according to their own organizational principals. This interface was designed to be the most flexible and provide the most organizationally complex user-view of the database.

3. Menu--The subject selected items from (usually) pre-defined menus. The upper-level menus followed a strict hierarchical organization and were generated in advance. At the leaf level, the set of items which comprised final menus were automatically generated by Virtual Interface System. But the important point is that the subjects view of the database was hierarchical with a limited, well-defined (and visually available to the subject) choice set at each level. Thus, this interface type was designed to represent an intermediate level of database organizational complexity.

Within all three interface types, command-set size was either large or small (i.e., the interface either comprised a small number of general-purpose commands or a large number of very specific,

TABLE 1

INTERFACE FEATURES

INTERFACE #	SEARCH MODE	JUMP MODE	BACK UP/RESTART	COMMAND SET
1	Key Word	—	—	Small
2	Key Word	—	—	Large
3	Sequential	—	—	Small
4	Sequential	—	—	Large
5	Menu	Single	yes	Small
6	Menu	Single	no	Small
7	Menu	Multiple	yes	Small
8	Menu	Multiple	no	Small
9	Menu	Single	yes	Large
10	Menu	Single	no	Large
11	Menu	Multiple	yes	Large
12	Menu	Multiple	no	Large

unique commands. Each command set—large or small—provided the same functionality.).

In addition, within the Menu interface type ONLY the following features were manipulated. (It will be seen that the following interface features do not "make sense" within the context of Keyword and Sequential interfaces. For example, a feature that does or does not allow "backup" to the previous menu level is meaningless in an interface that does not have choice levels.)

1. Backup+Restart/Restart Only: Four Menu interfaces provided a "backup" feature whereby the subject could return to the "next higher" menu level and make another choice. The other four interfaces provided only "restart" whereby the subject could restart a search by returning directly to the "top menu." (The interfaces with "backup" also provided "restart.")

2. Single/Multi-Jump: Four Menu interfaces provided a "multi-jump" feature whereby the subject could make multiple menu selections at one time. Thus, if the subject was at the top menu and was familiar with the content of second-level menus then the subject could make both first and second level menu choices at one time. This would bypass the second level menu presentation and "jump" the subject directly to the third menu level.

Experimental Task

The basic experimental task involved "meal planning." Four problems (consisting of four menus) were used for the experimental task. Problem 1 was defined as the training. During the course of the solution of this problem, the subjects went through all the steps to learn the interface he/she was assigned to. Within each problem there were subsegments of the problem. A typical problem, for example, was to "plan a dinner menu consisting of sea food as the main dish, two vegetables, some form of potato, and a dessert."

All the subjects were furnished with an instructional booklet. A sample of instructions is attached in Appendix C. In the

instruction booklet, a subject was given an example of how to retrieve an item of sea food as the main dish for this particular menu. The subject was told to read the instructions carefully, and feel free to ask any questions he/she may have. The same example was used for all the interfaces with alterations depending on a particular interface a subject was assigned to. The Subject was given a problem consisting of a "menu specification" such as "picnic lunch for 5 people." The subject had to find items in a 1500-item recipe database to generate a set of selections satisfying the problem demands.

In solving each problem, the subject had to find and retrieve items from the recipe database. When the subject found an item that they wanted to include in the problem solution set, the command "select" (or some variant, depending upon the particular interface) would be entered. If the selected item was appropriate for the problem, the message "Good Choice" would appear on the subject's VDU. Thus, the subject had to keep searching until a set of "Good Choice" items had been found.

Subjects were asked to write down their final set of choices on the note pad which was provided at their desk.

The complete set of problems is shown in the accompanying Appendix C.

Testing Procedures

Subjects were tested individually. The test session included two phases, training and information retrieval or problem-solving.

Training—Phase 1

During the training session, the subject worked through a practice problem according to a step-by-step sequence of instructions. The sequence was designed to introduce each command in the interface and to illustrate its use. This method ensured that the subject gained experience with each command and allowed the experimenter, who was observing from the E-computer monitor, to identify problems that subjects might have with particular commands. Since subjects had to complete each step in the instruction sequence to finish the problem properly, this method ensured that all commands had been executed correctly by the subject during training.

The problem used during training was similar in all respects to the type of problem that subjects actually had to solve during the main testing phase. Data recorded during the practice session was also identical to that collected during the second phase.

The training phase ended when the subject entered the command "done." (The "done" command was common to all interfaces. It was used by the subject to indicate that they felt a satisfactory problem solution had been reached and to indicate readiness for the next problem). The entire length of the training session was recorded.

Help Availability

If the subject appeared to be experiencing any difficulty or asked for help, the experimenter intervened and explained the operation of a command or concept. (It was felt that this represented an ecologically-valid situation: In learning to use a real computer application a new user would likely be able to ask

for help from a local expert. To avoid experimenter intervention would probably have created an unrealistically stressful situation. Williges and Williges (1985) have shown that availability of help impacts the nature of the interaction process.)

The instruction sequence was given to the subject on pages of paper. So all during the experiment the subject would be able to refer to examples of command usage. Also, a simple form of on-line help was available. By entering the command "?" or "help" the subject could see a list of command names along with an abbreviated command purpose/syntax description. This help command was available to all twelve interfaces. Whenever the subject viewed this help information, the length of the viewing was recorded.

The set of instructions given to subject's for each interface is available in the Appendix.

Main Problem-Solving Phase (or Information Retrieval)—Phase 2

The main problem-solving phase commenced immediately after the training phase. After the subject typed "done" to indicate conclusion of the training problem, the message "Please Wait..." appeared and the next problem was transmitted from the E-computer to the S-computer. The problem automatically appeared on the subject's VDU when it arrived. The subject could study the problem as long as desired. The prompt "Press any key when ready:" appeared at the bottom of the subject's VDU.

The subject could review the problem at any time by entering the command "problem." This command was common to all interfaces.

When the subject indicated readiness to begin by pressing a Key, the experimental phase started. An interaction screen

appropriate for the interface in effect appeared on the subject's VDU along with the prompt "enter command:" at the bottom of the display. So the "first move" was the subject's. The subject entered a command by entering a character string terminated by [RETURN].

When a command was entered, the message "Please wait..." appeared on the subject's VDU. The command was transmitted to the E-computer where the experimenter constructed an appropriate response and then sent the reply back to the S-computer. The response appeared automatically on the subject's VDU and the previously-entered command was cleared from the command line. This process continued until the subject achieved a "satisfactory" problem solution. A total of three problems was presented.

Throughout the experiment subjects had a pad of paper upon which to record notes and intermediate results and information. These notes were kept as part of the experiment's data output file.

EXPERIMENTAL DESIGN

Main Analysis

The interfaces included in this experiment have been generated by a combination of various factors. These factors are search modes, command set size, jump mode, and backup options. There are three search modes: keyword, sequential and menu; two command set sizes: large and small; two levels of jump mode: single/multiple jump; and two levels of backup: backup-yes/no. A full factorial set will consist of twenty-four combinations defined by three (search modes) x two (command sets) x two (groups) x two (backup levels). However, the jump mode and backup facility is not

applicable to keyword and sequential search modes, therefore, not all of the twenty-four configurations are feasible.

The experimental design used in this experiment, which incorporated all the feasible combinations of the interface configuration, comprised of three factors between subjects' design. These factors are search modes (six levels), command set size (two levels), groups (two levels). The six search mode levels were defined as follows:

SM1 = (Key Word Search)

SM2 = (Sequential Search)

SM3 = (Menu Search, Single Jump, Backup-No)

SM4 = (Menu Search, Single Jump, Backup-Yes)

SM5 = (Menu Search, Multiple Jump, Backup-No)

SM6 = (Menu Search, Multiple Jump, Backup-Yes)

The command set size was defined as small and large, and group levels were designated as field dependent (FD) and field independent (FI). The treatment combination formed by two factors--command set size and search mode--comprised the 12 interfaces used in the Experiment II (see Table 2).

Subject Assignment

Two groups of seventy-two subjects participated in the experiment. Seventy-two subjects in group one (FD) were randomly assigned to twelve interface conditions; assigning six subjects to each interface. The same procedure was utilized to assign subjects in group two (FI).

TABLE 2

Interface Conditions

	SEARCH MODE/JUMP/BACKUP	COMMAND SET
1.	SM1 (KW)	Small
2.	SM1 (KW)	Large
3.	SM2 (SEQ)	Small
4.	SM2 (SEQ)	Large
5.	SM3 (MENU)/SINGLE/YES	Small
6.	SM3 (MENU)/SINGLE/NO	Small
7.	SM4 (MENU)/MULTIPLE/YES	Small
8.	SM4 (MENU)/MULTIPLE/NO	Small
9.	SM5 (MENU)/SINGLE/YES	Large
10.	SM5 (MENU)/SINGLE/NO	Large
11.	SM6 (MENU)/MULTIPLE/YES	Large
12.	SM6 (MENU)/MULTIPLE/NO	Large

Dependent Variables

Experiment 2 consisted of two phases: training and information retrieval (or problem solution). For each of the two phases--phase 1: training, and phase 2: problem solution or information retrieval--the following dependent variables were considered. Training time (TRN), problem solution time (PST), total time (TT) for both training and information retrieval, number of commands used during training (CI), number of commands used during problem solution (TOTC), number of errors during training (EI), and total number of errors during problem solution/information retrieval (TOTE). Thus, a total of seven variables have been analyzed in the present analyses.

DEPENDENT VARIABLE

DEPENDENT VARIABLE	PHASE	ANALYSIS LEVEL
Training Time	1	TRN
Problem Solution Time (Information Retrieval)	2	PST
Total Number of Commands in Training	1	CI
Total Number of Commands Used During Information Retrieval	2	TOTC
Total Number of Errors During Training	1	EI
Total Number of Errors Made During Information Retrieval	2	TOTE
Total Time	(1&2)	TT

Experimental Design—Sub-Analysis—Menu Mode Only

It was pointed out in the preceding sections that the options such as Jump (multiple/single) and Backup (backup/restart) were applicable to menu mode only. Therefore, to assess whether the options play a role, if any, in Human Computer Interaction (HCI), a four factorial between subjects' design was used. These factors consisted of jump modes (two levels; single/multiple), backup (two levels; backup/restart), command set (two levels; small/large) and groups (two levels; FI/FD).

RESULTS

MAIN ANALYSIS—EXPERIMENT II

In this analysis, two main statistical techniques, Analysis of Variance and Multiple Comparison with the Best Treatment, were used for each of these seven variables. Some of these variables exhibited hetrogeneity of variance within some cells. In order to stabilize these variances, analyses based on log transformation were also conducted. However, the analysis based on transformed data were qualitatively identical to that based on the original data. Therefore, only the results based on original data are presented in this report.

Training Time (TRN)

The analysis of variance results for training time are shown in Table 3. Since the main hypotheses in this project are the interaction effects, therefore, only the interactions will be evaluated in this discussion. All the interactions involving search modes, command set size, and groups were significant. However, detailed explanation for interactions involving groups will be presented here.

First, the three way interaction of group by size by search mode was significant $F(5,120)=2.97$ ($p=.0146$). This interaction is due to the fact that field independent subjects consistently spent less time while using small command set (size 1) to retrieve target items, and they spent more time to solve the same problem when presented with an option of large command set, i.e., function specific commands (see Table 4). However, this trend was not true

for field dependent subjects. Field dependent subjects' target retrieval time varied depending on the search mode and the option combinations. For example, in key word search mode they spent more time when using small command set and less time when using large command set. When they were presented with sequential mode they spent less time while using small command set and more time when using large command set. Looking at Menu search modes, (SM4 - SM6) when the subjects were presented with the option of restart (SM3 & SM5) their target item retrieval time was less with small command set size and more while using large command set size. However, when presented with backup option, they spent more time to retrieve target items with small command set and less time while using large command set.

A highly significant two way interaction is group by search mode; $F(5,120)=33.0$, ($p=.0001$). The results clearly indicate that a user's performance depends on the interface design and his/her natural mode of processing information (Table 5 & 6).

The mean time for FI subjects when using KW search mode is 28.25 minutes and 53.36 minutes when using sequential search mode, the subjects took 53.36 minutes on the average to solve a problem. The reverse was true for field dependent subjects. They showed strong preference for sequential search mode. Looking at Table 6, when field dependent subjects were presented with their preferred sequential mode, they took 24.22 minutes to solve the problem. However, when presented with non-preferred mode, they used 50.59 minutes to solve the problem. These findings clearly reveal that the design of the interface plays a significant role in an individual's performance during information retrieval task.

Another significant two way interaction was group by size $F(1,120)=9.78$ ($p=.0022$). Table 7 shows, overall, FI subjects spent relatively less time when using small command set and more time with the option of large command set. FD subjects, on the other hand, spent more time when using small command set and less time when using large command set size.

In summary, the results reveal that FI subjects showed strong preference for command set size which they could manipulate or restructure, e.g., combine commands to reduce the steps to retrieve the target item. They showed preference for search modes—KW and menu with multiple jump and backup, whereas FD subjects preferred command sets which required the least amount of restructuring or manipulation and preferred sequential search mode. These results are consistent with theoretical framework of psychological differentiation (Witkin, 1962) that FD subjects are context bound, therefore, are better at tasks which are structured, whereas FI subjects are flexible, therefore can restructure the tasks as they find fit.

Problem Solution Time (PST)

The analysis of variance results for PST are shown in Table 8. The interaction effect of group by search mode was also highly significant for this variable $F(5,120)=30.84$, ($p=.0001$). FI subjects showed strong preference for search mode types which are flexible such as KW and their least preferred mode was sequential highly structured mode as shown in Table 5. FI subjects, on the average took 29 minutes and 40 seconds to retrieve target items to solve a problem, when presented with KW mode and 1 hour, 4 minutes

TABLE 3

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: TRN

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	23	26833.7929	1166.6867	16.25	0.0001	0.75569	29.6136
Error	120	8617.0040	71.8084		Root Mse		TRN Mean
Corrected Total	143	35450.7969			8.4740		28.6151

Source	DF	Type I SS	F Value	Pr>F
Grp	1	3829.2891	53.33	0.0001
Size	1	10.6978	0.15	0.7002
Grp*Size	1	702.2014	9.78	0.0022
M	5	7541.7252	21.01	0.0001
Grp*M	5	11849.1778	33.00	0.0001
Size*M	5	1834.4146	5.11	0.0003
Grp*Size*M	5	1066.2870	2.97	0.0146

ANALYSIS OF VARIANCE RESULTS—TRAINING
GROUPS BY COMMAND SET SIZE BY SEARCH MODES

TABLE 4

SAS GRP=1

TRN		M					
		1	2	3	4	5	6
Size							
1	Mean	26.62	46.61	12.67	10.52	16.97	15.74
	Std	6.31	14.57	5.71	2.82	2.53	4.35
2	Mean	29.07	59.30	18.40	11.85	17.42	16.31
	Std	5.78	27.20	2.27	1.18	3.43	1.84

SAS GRP=2

TRN		M					
		1	2	3	4	5	6
Size							
1	Mean	50.99	22.66	31.47	48.58	25.86	37.95
	Std	5.05	4.17	6.98	13.17	2.86	12.97
2	Mean	50.19	24.97	36.37	20.67	30.66	24.88
	Std	7.46	3.24	4.07	7.84	4.82	1.16

MEAN AND STANDARD DEVIATION FIGURES FOR TRAINING GROUPS BY COMMAND SET SIZE BY SEARCH MODES

TABLE 5

GRP=1
M

	1		2		3	
	Mean	Std	Mean	Std	Mean	Std
TT	50.25	6.65	116.84	38.17	45.95	15.21
TRN	27.85	5.91	52.96	21.84	15.53	5.11
PST	22.41	5.13	63.89	25.60	30.41	11.55
TOTC	49.67	9.46	60.83	18.71	90.50	30.53
C1	22.92	7.93	47.33	19.03	29.83	7.15
E1	3.17	2.86	6.50	6.72	2.92	2.35
TOTE	4.17	2.98	5.58	8.14	5.42	5.33

GRP=1
M

	4		5		6	
	Mean	Std	Mean	Std	Mean	Std
TT	34.12	9.67	42.13	6.61	38.59	8.31
TRN	11.19	2.18	17.20	2.88	16.02	3.20
PST	22.94	8.27	24.94	6.33	22.57	7.95
TOTC	73.92	13.47	74.25	9.23	83.42	14.03
C1	28.08	10.57	27.33	2.10	26.17	7.33
E1	3.67	6.17	2.25	2.34	3.08	2.71
TOTE	4.83	6.44	6.42	4.36	6.00	4.67

GROUP 1: MEANS AND STANDARD DEVIATION FIGURES
FOR SEARCH MODES FOR ALL THE DEPENDENT VARIABLES

TABLE 6

GRP=2
M

	1		2		3	
	Mean	Std	Mean	Std	Mean	Std
TT	117.13	21.12	52.52	6.56	73.47	8.41
TRN	50.59	6.90	23.82	3.76	33.92	6.02
PST	66.54	16.76	28.70	6.15	39.55	6.95
TOTC	95.42	12.41	29.92	3.37	102.33	22.09
C1	30.92	7.55	25.82	5.72	41.67	6.72
E1	3.33	2.74	2.83	3.51	5.58	7.37
TOTE	19.83	7.21	1.17	1.03	6.92	3.37

GRP=2
M

	4		5		6	
	Mean	Std	Mean	Std	Mean	Std
TT	76.69	25.84	69.44	8.66	76.59	16.18
TRN	34.63	17.86	28.26	4.54	31.41	11.12
PST	42.06	11.26	41.18	6.66	45.18	9.11
TOTC	95.42	15.64	89.42	10.47	93.17	13.00
C1	46.17	12.35	44.92	11.61	45.58	11.46
E1	4.58	2.81	2.17	1.47	2.00	2.17
TOTE	4.42	2.27	3.17	2.79	3.92	2.07

GROUP 2: MEANS AND STANDARD DEVIATION FIGURES
FOR SEARCH MODES FOR ALL THE DEPENDENT VARIABLES

TABLE 7

SAS GRP=1

	Size 1		Size 2	
	Mean	Std	Mean	Std
TT	53.72	30.85	55.57	36.07
TRN	21.52	14.19	25.39	19.44
PST	32.20	20.32	30.18	18.72
TOTC	65.44	15.22	78.75	25.15
C1	27.78	11.58	32.78	13.59
E1	3.58	4.63	3.61	4.11
TOTE	5.69	5.73	5.11	5.15

SAS GRP=2

	Size 1		Size 2	
	Mean	Std	Mean	Std
TT	79.75	24.32	75.53	25.92
TRN	36.25	13.45	31.29	11.09
PST	43.49	12.93	44.24	17.13
TOTC	84.19	28.95	84.36	27.89
C1	42.56	11.98	35.81	11.65
E1	4.44	5.03	2.39	1.99
TOTE	7.03	7.61	6.11	6.76

MEAN AND STANDARD DEVIATION FIGURES FOR GROUP
BY COMMAND SET SIZE FOR ALL THE DEPENDENT VARIABLES

TABLE 8

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: PST

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	23	32619.2250	1418.2272	10.58	0.0001	0.6697	30.8516
Error	120	16087.1835	134.0599		Root Mse		PST Mean
Corrected Total	143	48706.4085			11.5784		37.5294

Source	DF	Type I SS	F Value	Pr>F
Grp	1	5785.4278	43.16	0.0001
Size	1	14.6715	0.11	0.7414
Grp*Size	1	68.9730	0.51	0.4746
M	5	4565.6757	6.81	0.0001
Grp*M	5	20674.2698	30.84	0.0001
Size*M	5	849.1693	1.27	0.2821
Grp*Size*M	5	661.0378	0.99	0.4299

ANALYSIS OF VARIANCE RESULTS FOR PROBLEM SOLUTION TIME (PST)
THE FACTORS INCLUDED GROUPS, COMMAND SET SIZE, AND SEARCH MODES

and 29 seconds while using the sequential search mode.

The situation was reversed for field dependent subjects in terms of their search mode preference. Again, looking at Table 6, we can see when field dependent subjects were faced with their non-preferred mode, that is, key word, their performance time was 66 minutes and 54 seconds, but when presented with their preferred mode, the time they took was only 29 minutes and 10 seconds.

The pattern of these results are identical to the results evidenced in training.

Total Time (TT)

The analysis of variance results for this variable are shown in Table 9. Basically, the results are consistent with training and problem solution time. Again, the interaction of group by search mode is significant, $F(5,120)=46.46$, ($p=.0001$). Looking at Table 6, it is clear that FD subjects showed strong preference for sequential mode as it proved to be the most efficient search mode for them. On the average, the total amount of time spent to solve all the problems was 52 minutes and 50 seconds for their preferred mode, they spent 117 minutes and 13 seconds for their nonpreferred mode. This finding is again consistent with training phase and problem solution time.

Another significant interaction was command set size by mode $F(5,120)=3.17$, ($p=.0102$) which is shown in Table 10. Subjects differentiated in time usage as a function of command set size. This finding is consistent with training time. The consistency of these results reveal that the performance is significantly influenced by a combination of interface design and user's

information processing style. The three way interaction of group by size by mode did not prove to be significant.

Number of Commands Used During Training Session (C1)

The analysis of variance results for the variable C1 are shown in Table 11. For this variable, the three factor interaction of group by size by mode is not significant. However, all two way interactions are significant. First highly significant two way interaction again is group by search mode $F(1,120)=18.27$, ($p=.0001$). The results shown in Table 12 indicate that FI subjects on the average used fewer commands for keyword search mode relative to sequential search mode, however, this command usage trend was not consistent with FD subjects. They used minimum number of commands for sequential mode as compared to keyword and menu mode.

Another significant interaction associated with C1 is size by mode $F(5,120)=6.04$, ($p=.01$) and group by search mode. Looking at Table 12, we can see that field independent subjects used maximum number of commands for their nonpreferred mode, specifically more for large command set size than small command set size. This trend was consistent for all their preferred mode relatively less for large command set size than small command set size, this trend of using more commands for small command set than large command set was consistent with almost all the search modes, which is basically the opposite of the pattern observed with field independent participants.

Number of Commands Used During Problem Solution Phase (TOTC)

Another dependent variable analyzed is number of commands used during problem solution. The results of analysis of variance are

TABLE 9

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: TT							
Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	23	110873.2949	4820.5780	18.30	0.0001	0.7782	24.5360
Error	120	31606.5059	263.3876		Root Mse		TT Mean
Corrected Total	143	142479.8008			16.2292		66.1445

Source	DF	Type I SS	F Value	Pr>F
Grp	1	19028.3402	72.24	0.0001
Size	1	50.4254	0.19	0.6625
Grp*Size	1	331.0247	1.26	0.2645
M	5	23730.5159	18.02	0.0001
Grp*M	5	61183.0397	46.46	0.0001
Size*M	5	4171.1097	3.17	0.0102
Grp*Size*M	5	2378.8394	1.81	0.1156

ANALYSIS OF VARIANCE RESULTS FOR TOTAL TIME (TT)
THE FACTORS INCLUDED GROUPS, COMMAND SET SIZE, AND SEARCH MODES

TABLE 10

SAS GRP=1

TT		M					
		1	2	3	4	5	6
Size							
1	Mean	50.11	114.22	37.01	35.98	44.66	40.38
	Std	6.16	27.50	13.00	12.27	4.14	9.93
2	Mean	50.39	119.47	54.89	32.27	39.61	36.81
	Std	7.71	43.32	12.17	6.85	7.97	6.76

SAS GRP=2

TT		M					
		1	2	3	4	5	6
Size							
1	Mean	112.75	51.87	70.52	96.23	64.19	82.93
	Std	12.48	8.52	8.02	22.73	2.93	18.77
2	Mean	121.51	53.18	76.42	57.15	74.68	70.25
	Std	27.93	4.58	8.39	6.01	9.51	11.30

MEAN AND STANDARD DEVIATION FIGURES FOR TOTAL TIME (TT)
FOR GROUP BY SIZE BY SEARCH MODES

TABLE 11

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: C1

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	23	15572.9375	677.0842	8.56	0.0001	0.6214	25.6030
Error	120	9487.5000	79.0625		Root Mse		C1 Mean
Corrected Total	143	25060.4375			8.8917		34.7292

Source	DF	Type I SS	F Value	Pr>F
Grp	1	2853.3403	36.09	0.0001
Size	1	27.5625	0.35	0.5560
Grp*Size	1	1242.5625	15.72	0.0001
M	5	1788.3958	4.52	0.0009
Grp*M	5	7223.4514	18.27	0.0001
Size*M	5	2386.3958	6.04	0.0001
Grp*Size*M	5	51.2292	0.13	0.9832

ANALYSIS OF VARIANCE RESULTS FOR NUMBER OF COMMANDS USED DURING TRAINING (C1)
THE FACTORS INCLUDED GROUPS, COMMAND SET SIZE, AND SEARCH MODES

TABLE 12

SAS GRP=1

C1		M					
		1	2	3	4	5	6
Size							
1	Mean	17.17	44.50	26.00	19.33	28.50	31.17
	Std	6.65	12.63	8.58	6.80	2.43	5.95
2	Mean	28.67	50.17	33.67	36.83	26.17	21.17
	Std	3.83	24.86	1.86	3.97	0.75	4.79

SAS GRP=2

C1		M					
		1	2	3	4	5	6
Size							
1	Mean	31.50	29.33	45.50	43.67	50.50	54.83
	Std	7.18	5.75	6.22	13.41	7.40	4.71
2	Mean	30.33	22.33	37.83	48.67	39.33	36.33
	Std	8.55	3.08	5.04	11.86	12.93	7.84

MEAN AND STANDARD DEVIATION FIGURES FOR C1 GROUPS BY COMMAND SET SIZE AND SEARCH MODES

shown in Table 13. Again, only the interaction effects will be discussed here. First, in contrast to variable C1, the three factor interaction of group by size by search mode was highly significant $F(5,120)=3.85$, ($p=.0030$). The most important performance pattern among FD subjects was that they consistently showed efficiency in command usage for sequential search mode (see Table 14). There does not seem to be much difference between usage of small command set and large command set for menu and KW search mode among FD subjects.

Another significant interaction revealed was group by mode $F(5,120)=18.30$, ($p=.0001$). The pattern of results is similar to what was observed in time variables.

Again, FI and FD subjects showed differential preference for search mode types. Looking at Table 5, it is clear that field independent subjects utilized less commands for KW than sequential search mode. For example, FI subjects, on the average, used 49.67 commands to solve problem of retrieving all the target items using KW versus 60 commands when sequential search mode. The search mode preference was reversed for FD subjects. On the average, they used 95 commands for KW and 23 for sequential mode. Therefore, the conclusion can be drawn in terms of compatibility between interface design and the user. If the interface design is compatible with the user's information processing style, they use less commands, and if the interface design in incompatible with the user's information style, they use more commands. These results again are consistent with variables involving time.

TABLE 13

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: TOTC

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	23	71054.4375	3089.3234	15.27	0.0001	0.7453	18.1947
Error	120	24285.5000	202.3792		Root Mse		TOTC Mean
Corrected Total	143	95339.9375			14.2260		78.1875

Source	DF	Type I SS	F Value	Pr>F
Grp	1	5341.1736	26.39	0.0001
Size	1	1633.5069	8.07	0.0053
Grp*Size	1	1553.6736	7.68	0.0065
M	5	38356.8958	37.91	0.0001
Grp*M	5	18516.4514	18.30	0.0001
Size*M	5	1752.9514	1.73	0.1312
Grp*Size*M	5	3899.7847	3.85	0.0030

ANALYSIS OF VARIANCE RESULTS FOR NUMBER OF COMMANDS USED DURING PROBLEM SOLUTION TIME (TOTC)
THE FACTORS INCLUDED GROUPS, COMMAND SET SIZE, AND SEARCH MODES

TABLE 14

SAS GRP=1

TOTC

		M					
		1	2	3	4	5	6
Size							
1	Mean	49.17	58.00	67.00	63.33	68.50	86.67
	Std	11.89	6.45	8.56	9.09	9.18	15.95
2	Mean	50.17	63.67	114.00	84.50	80.00	80.17
	Std	7.41	26.64	25.53	6.89	4.86	12.38

SAS GRP=2

TOTC

		M					
		1	2	3	4	5	6
Size							
1	Mean	94.83	29.50	105.17	96.33	89.50	89.83
	Std	13.93	3.08	24.47	17.78	13.22	8.33
2	Mean	96.00	30.33	99.50	94.50	89.33	96.50
	Std	11.98	3.88	21.35	14.83	8.14	16.61

MEAN AND STANDARD DEVIATION FIGURES FOR TOTC
GROUPS BY SEARCH MODES BY COMMAND SET SIZE

SUB-ANALYSIS--MENU MODE ONLY

Rationale:

This analysis was conducted for several reasons.

1. In Experiment 1, the two search modes considered were Key word and sequential only. In order to check all the pertinent interface design combinations in the current experiment, menu mode was included.

2. The options such as backup (yes/restart) jump (multiple versus single) are applicable only to menu search mode and the effect of these combinations were not determined in the main analysis.

3. Menu search mode is used in a wide range of database search, it was felt necessary to check if FI and FD users show any different preferential treatment for menu mode with or without options.

Sub-Analysis Results

This sub-analyses is based on a four factor between subject experimental model. The dependent variables are again Training time (TRN), Problem solution time (PST), Total time (TT), C1 (Total number of commands used during training), TOTC (Total number of commands used during problem solution).

Training Time (TRN)

As indicated in the main analysis, only the interaction effect involving groups will be considered. The results of the analysis of variance are shown in Table 15.

The most important finding from interface design point of view

TABLE 15

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: TRN

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	15	10635.7361	709.0491	19.27	0.0001	0.7832	25.7895
Error	80	2943.6659	36.7958		Root Mse		TRN Mean
Corrected Total	95	13579.4020			6.0660		23.5211

Source	DF	Type I SS	F Value	Pr>F
Grp	1	6993.8009	190.07	0.0001
Size	1	201.6573	5.48	0.0217
Grp*Size	1	581.0077	15.79	0.0002
Jm	1	8.4461	0.23	0.6332
Grp*Jm	1	354.5283	9.64	0.0026
Size*Jm	1	28.4371	0.77	0.3820
Grp*Size*Jm	1	162.1022	4.41	0.0390
Bup	1	4.1371	0.11	0.7383
Grp*Bup	1	132.0071	3.59	0.0618
Size*Bup	1	1131.4413	30.75	0.0001
Grp*Size*Bup	1	807.2382	21.94	0.0001
Jm*Bup	1	47.4455	1.29	0.2595
Grp*Jm*Bup	1	0.7973	0.02	0.8833
Size*Jm*Bup	1	142.0896	3.86	0.0529
Grp*Size*Jm*Bup	1	40.6003	1.10	0.2967

ANALYSIS OF VARIANCE RESULTS FOR SUB-ANALYSIS FOR TRAINING—
MENU SEARCH MODE ONLY—GROUPS BY COMMAND SET SIZE BY JUMP BY BACKUP

is a highly significant three way interaction involving group by size by backup, $F(1,80)=21.94$, ($p=.0001$). FI subjects preferred the option combinations of small command set and backup. In other words they spend less time to solve the problem relative to the option of large command set and backup (see Table 16). Furthermore, the worst performance for FI subjects was when the option combination was small command set and restart. The rationale being that this combination is relatively structured (within menu mode) and does not have much room for manipulation, therefore, is not compatible with FI subjects mode of processing information. Field dependent subjects' performance was best when the option combination was backup and large command set. Looking at Table 16 again, comparing backup and restart, FD subjects showed strong preference for restart option.

Another three way interaction involving group by size by jump was also significant, $F(1,80)=4.41$ ($p=.0390$). When the option of jump (single/multiple) was combined with command set size (small/large), the FI subjects spent less time with single jump and large command set and more time with single jump and small command set. FD subjects showed preference for large command set and single jump. They spent less time to retrieve the target items to solve the problem using this combination. Furthermore, preference was shown for multiple jump and large command set compared to multiple jump and small command set (See Table 17).

Another highly two way significant interaction was group by command set size, $F(1,80)=15.79$, ($p=.0002$). The results indicate that FI and FD subjects showed differential preference for small/large command set size. FD subjects spend less time to

TABLE 16

SAS GRP=1

TRN		Size 1	Size 2
BUP			
1	Mean	13.13	14.08
	Std	4.43	2.76
2	Mean	14.82	17.91
	Std	4.77	2.82

SAS GRP=2

TRN		Size 1	Size 2
BUP			
1	Mean	43.26	22.78
	Std	13.64	5.78
2	Mean	28.67	33.51
	Std	5.87	5.19

MEAN AND STANDARD DEVIATION FIGURES FOR TRN—MENU MODE ONLY
GROUPS BY COMMAND SET SIZE BY BACKUP

TABLE 17

SAS GRP=1

TRN		Size 1	2
JM			
1	Mean	11.59	15.13
	Std	4.44	3.83
2	Mean	16.36	16.87
	Std	3.45	2.59

SAS GRP=2

TRN		Size 1	2
JM			
1	Mean	40.03	28.52
	Std	13.44	10.13
2	Mean	31.90	27.77
	Std	10.96	4.50

MEAN AND STANDARD DEVIATION FIGURES FOR TRN--MENU MODE ONLY
GROUPS BY COMMAND SET SIZE BY JUMP (SINGLE/MULTIPLE)

retrieve items to solve problems when utilizing large command set and they spend more time when using small command set. The opposite was true for FI subjects. They spent less time when given the option of small command set (See Table 18). The rationale for the results is consistent with the hypothesis of FD subjects being context bound. FD subjects preferred the command set which required least amount of restructuring and FI subjects portrayed preference for the small command set size because they could restructure it.

Another two factor significant interaction is group by jump $F(1,80)=9.64$, ($p=.0026$). FD subjects showed preference for single jump whereas FI subjects preferred multiple jump option (Table 19).

Problem Solution Time (PST)

Second variable under investigation was PST. The results of the analysis of variance are shown in Table 20. Only two interactions involving groups were significant. First significant interaction was groups by command set size by jump, $F(1,80)=5.80$ ($p=.0174$), FI subjects preferred the option combination of small command set with single jump, whereas FD subjects preferred the option combination of large command set and single jump (See Table 21). These results are consistent with TRN variable.

Another significant interaction is group by backup, $F(1,80)=5.80$, ($p=.0184$), indicating that field independent and field dependent subjects differed as a function of backup versus restart function. Looking at Table 22, FI subjects used less time with backup option and field dependent subjects spent less time when presented with the option of restart. These results are again

retrieve items to solve problems when utilizing large command set and they spend more time when using small command set. The opposite was true for FI subjects. They spent less time when given the option of small command set (See Table 18). The rationale for the results is consistent with the hypothesis of FD subjects being context bound. FD subjects preferred the command set which required least amount of restructuring and FI subjects portrayed preference for the small command set size because they could restructure it.

Another two factor significant interaction is group by jump $F(1,80)=9.64$, ($p=.0026$). FD subjects showed preference for single jump whereas FI subjects preferred multiple jump option (Table 19).

Problem Solution Time (PST)

Second variable under investigation was PST. The results of the analysis of variance are shown in Table 20. Only two interactions involving groups were significant. First significant interaction was groups by command set size by jump, $F(1,80)=5.80$ ($p=.0174$), FI subjects preferred the option combination of small command set with single jump, whereas FD subjects preferred the option combination of large command set and single jump (See Table 21). These results are consistent with TRN variable.

Another significant interaction is group by backup, $F(1,80)=5.80$, ($p=.0184$), indicating that field independent and field dependent subjects differed as a function of backup versus restart function. Looking at Table 22, FI subjects used less time with backup option and field dependent subjects spent less time when presented with the option of restart. These results are again

TABLE 18

SAS GRP=1

	Size 1		Size 2	
	Mean	Std	Mean	Std
TRN	13.97	4.59	16.00	3.36
TT	39.50	10.33	40.89	11.89
PST	25.53	7.36	24.90	10.59
TOTC	71.38	13.87	89.67	19.99
C1	26.25	7.43	29.46	6.99
E1	2.63	2.50	3.33	4.58
TOTE	6.08	5.56	5.25	4.74

SAS GRP=2

	Size 1		Size 2	
	Mean	Std	Mean	Std
TRN	35.97	12.69	28.15	7.68
TT	78.47	19.02	69.63	11.40
PST	42.50	8.72	41.48	8.76
TOTC	95.21	17.15	94.96	15.35
C1	48.63	9.17	40.54	10.48
E1	4.33	5.81	2.83	1.74
TOTE	4.96	3.13	4.25	2.79

MEAN AND STANDARD DEVIATION FIGURES FOR ALL THE DEPENDENT VARIABLES GROUPS BY SIZE—MENU MODE ONLY

TABLE 19

SAS GRP=1

	Jm 1		2	
	Mean	Std	Mean	Std
TRN	13.36	4.44	16.61	3.04
TT	40.04	13.85	40.36	7.56
PST	26.67	10.54	23.75	7.13
TOTC	82.21	24.58	78.83	12.52
C1	28.96	8.87	26.75	5.31
E1	3.29	4.58	2.67	2.51
TOTE	5.13	5.79	6.21	4.42

SAS GRP=2

	Jm 1		2	
	Mean	Std	Mean	Std
TRN	34.27	13.04	29.84	8.46
TT	75.08	18.86	73.01	13.21
PST	40.81	9.24	43.18	8.06
TOTC	98.88	19.05	91.29	11.70
C1	43.92	9.99	45.25	11.29
E1	5.08	5.48	2.08	1.82
TOTE	5.67	3.09	3.54	2.43

MEAN AND STANDARD DEVIATION FIGURES FOR ALL THE DEPENDENT VARIABLES GROUPS BY JUMP (SINGLE/MULTIPLE)—MENU MODE ONLY

TABLE 20

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: PST

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	15	8564.9605	570.9974	8.24	0.0001	0.6071	24.7740
Error	80	5543.8510	69.2981		Root Mse		PST Mean
Corrected Total	95	14108.8115			8.3246		33.6020

Source	DF	Type I SS	F Value	Pr>F
Grp	1	6754.9403	97.48	0.0001
Size	1	16.3556	0.24	0.6284
Grp*Size	1	0.9116	0.01	0.9090
Jm	1	1.8318	0.03	0.8713
Grp*Jm	1	168.0554	2.43	0.1234
Size*Jm	1	0.0965	0.00	0.9703
Grp*Size*Jm	1	408.4534	5.89	0.0174
Bup	1	16.6708	0.24	0.6251
Grp*Bup	1	401.6748	5.80	0.0184
Size*Bup	1	415.4792	6.01	0.0164
Grp*Size*Bup	1	1.0064	0.01	0.9044
Jm*Bup	1	65.2163	0.94	0.3349
Grp*Jm*Bup	1	19.7191	0.28	0.5952
Size*Jm*Bup	1	242.4274	3.50	0.0651
Grp*Size*Jm*Bup	1	51.1219	0.74	0.3930

ANALYSIS OF VARIANCE RESULTS FOR PROBLEM SOLUTION TIME (PST)—
MENU MODE ONLY—GROUP BY COMMAND SET SIZE BY JUMP BY BACKUP

TABLE 21

SAS GRP=1

PST		Size 1	2
JM			
1	Mean	24.90	28.45
	Std	8.43	12.43
2	Mean	26.16	21.34
	Std	6.43	7.23

SAS GRP=2

PST		Size 1	2
JM			
1	Mean	43.35	38.26
	Std	11.12	6.39
2	Mean	41.65	44.70
	Std	5.83	9.84

MEAN AND STANDARD DEVIATION FIGURES FOR PST
GROUPS BY JUMP BY COMMAND SET SIZE—MENU MODE ONLY

TABLE 22

SAS GRP=1

	BUP 1		BUP 2	
	Mean	Std	Mean	Std
TRN	13.61	3.64	16.37	4.15
TT	36.36	9.11	44.04	11.63
PST	22.75	7.93	27.68	9.53
TOTC	78.67	14.30	82.38	23.56
C1	27.13	8.95	28.58	5.31
E1	3.38	4.67	2.58	2.32
TOTE	5.42	5.53	5.92	4.79

SAS GRP=2

	BUP 1		BUP 2	
	Mean	Std	Mean	Std
TRN	33.02	14.64	31.09	5.96
TT	76.64	21.09	71.45	8.60
PST	43.62	10.14	40.36	6.71
TOTC	94.29	14.11	95.88	18.15
C1	45.88	11.66	43.29	9.42
E1	3.29	2.79	3.88	5.48
TOTE	4.17	2.14	5.04	3.58

MEAN AND STANDARD DEVIATION FIGURES FOR ALL THE VARIABLES
GROUPS BY BACKUP--MENU MODE ONLY

TABLE 23

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: TT

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	15	35445.5791	2363.0386	18.85	0.0001	0.7794	19.6021
Error	80	10030.4167	125.3802		Root Mse		TT Mean
Corrected Total	95	45475.9959			11.1973		57.1231

Source	DF	Type I SS	F Value	Pr>F
Grp	1	27495.4074	219.30	0.0001
Size	1	332.8735	2.65	0.1072
Grp*Size	1	627.9481	5.01	0.0280
Jm	1	18.1447	0.14	0.7046
Grp*Jm	1	34.4018	0.27	0.6019
Size*Jm	1	25.2212	0.20	0.6550
Grp*Size*Jm	1	1085.1864	8.66	0.0043
Bup	1	37.4176	0.30	0.5864
Grp*Bup	1	994.2204	7.93	0.0061
Size*Bup	1	2920.8323	23.30	0.0001
Grp*Size*Bup	1	865.2485	6.90	0.0103
Jm*Bup	1	223.9132	1.79	0.1852
Grp*Jm*Bup	1	28.4469	0.23	0.6351
Size*Jm*Bup	1	755.7119	6.03	0.0163
Grp*Size*Jm*Bup	1	0.6055	0.00	0.9448

ANALYSIS OF VARIANCE RESULTS FOR TOTAL TIME
GROUP BY COMMAND SET SIZE BY JUMP BY BACKUP--MENU MODE ONLY

TABLE 24

SAS GRP=1

TT		Size 1	Size 2
JM			
1	Mean	36.49	43.58
	Std	12.06	15.11
2	Mean	42.52	38.21
	Std	7.59	7.20

SAS GRP=2

TT		Size 1	Size 2
JM			
1	Mean	83.37	66.79
	Std	21.08	12.23
2	Mean	73.56	72.47
	Std	16.12	10.22

MEAN AND STANDARD DEVIATION FIGURES FOR TOTAL TIME GROUPS BY COMMAND SET SIZE BY JUMP—MENU MODE ONLY

TABLE 25

SAS GRP=1

TT		Size 1	Size 2
BUP			
1	Mean	38.18	34.54
	Std	10.89	6.91
2	Mean	40.83	47.25
	Std	10.03	12.65

SAS GRP=2

TT		Size 1	Size 2
BUP			
1	Mean	89.58	63.70
	Std	21.05	11.01
2	Mean	67.35	75.55
	Std	6.64	8.60

MEAN AND STANDARD DEVIATION FIGURES FOR TOTAL TIME GROUPS BY COMMAND SET SIZE BY BACKUP—MENU MODE ONLY

spent almost equal amount of time with large and small command set, whereas FD subjects spent significantly less time using large command set compared to small command set. Under the restart option, both the FI and FD subjects used equally more time with large command set as compared to small command set. Overall, FI subjects spent more time when option combination was large command set and restart and FD subjects spent significantly large amounts of time when the option combination was small command set and backup, indicating the consistency with the theoretical implications of psychological differentiation (Witkin, 1962) and its application to information retrieval tasks.

The significant two factor interaction shown in Table 23 is group by size $F(1,80)=5.01$, ($p=.0280$). The results indicate that the FI subjects spent relatively less time while using small command set as compared to large command set to retrieve all target items to solve the problem. The reverse was true with FD subjects, they spend less time when using large command set and more time while using large command set. These results are consistent with the variables TRN and PST discussed in preceding sections.

Total Number of Commands Used During Training Phase (C1)

The results of the analyses of variance for C1 are shown in Table 26. The significant interactions shown in Table 26 are size by jump by backup $F(1,80)=9.44$, ($p=.002$), group by command set size $F(1,80)=13.69$, ($p=.0004$) and jump by command size $F(1,80)=27.92$, ($p=.0001$). However, only the two factor interaction of group by size will be discussed here.

First, the group by size interaction has been consistently

TABLE 26

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: C1

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	15	10010.9063	667.3938	11.95	0.0001	0.6913	20.6372
Error	80	4469.5000	55.8688		Root Mse		C1 Mean
Corrected Total	95	14480.4063			7.4745		36.2188

Source	DF	Type I SS	F Value	Pr>F
Grp	1	6716.7604	120.22	0.0001
Size	1	142.5938	2.55	0.1141
Grp*Size	1	765.0104	13.69	0.0004
Jm	1	4.5938	0.08	0.7730
Grp*Jm	1	75.2604	1.35	0.2492
Size*Jm	1	1560.0938	27.92	0.0001
Grp*Size*Jm	1	41.3438	0.74	0.3922
Bup	1	7.5938	0.14	0.7133
Grp*Bup	1	98.0104	1.75	0.1891
Size*Bup	1	21.0938	0.38	0.5407
Grp*Size*Bup	1	3.7604	0.07	0.7960
Jm*Bup	1	15.8438	0.28	0.5958
Grp*Jm*Bup	1	29.2604	0.52	0.4714
Size*Jm*Bup	1	527.3438	9.44	0.0029
Grp*Size*Jm*Bup	1	2.3438	0.04	0.8382

ANALYSIS OF VARIANCE RESULTS FOR THE NUMBER OF COMMANDS USED DURING TRAINING (C1)
GROUP BY COMMAND SET SIZE BY JUMP BY BACKUP—MENU MODE ONLY

significant for all the variables evaluated thus far. The results clearly indicate FI subjects use less commands when using small command set--specialized function commands and FD subjects use more commands when using large command set--function specific commands. These results are consistent with prior findings of time variables.

Number of Commands Used During Problem Solution Phase (TOTC)

Another dependent variable evaluated was TOTC. The results of the analysis are shown in Table 27.

First three factor significant interaction is group by size by jump, $F(1,80)=4.09$ ($p=.0021$). Table 28 shows that FI subjects used less commands when the option combination was small command set and single jump versus large command set and single jump. FD subjects used less commands when the option combination was large command set and single jump, and they used significantly more commands when the option combination was small command set and single jump. When we look at the result using option combination of multiple jump and command set size, FI subjects used less commands when the option combination was large command set compared to the option combination of small command set and multiple jump. FD subjects did not differ significantly between the option combination of multiple jump and small command set and multiple jump and large command set.

Another significant three factor interaction is group by backup by size $F(1,80)=5.02$, ($p=.0278$). Looking at Table 29, FI subjects used less commands when the option combination was small command set and backup compared to option combination of large command set and backup. FD subject did not show much difference in

TABLE 27

SAS
GENERAL LINEAR MODELS PROCEDURE

Dependent Variable: TOTC

Source	DF	Sum of Squares	Mean Square	F Value	Pr>F	R-Square	C.V.
Model	15	17176.4063	1145.0938	5.16	0.0001	0.4920	16.9585
Error	80	17736.8333	221.7104		Root Mse		TOTC Me
Corrected Total	95	34913.2396			14.8899		87.8021

Source	DF	Type I SS	F Value	Pr>F
Grp	1	5089.5938	22.96	0.0001
Size	1	1953.0104	8.81	0.0040
Grp*Size	1	2062.7604	9.30	0.0031
Jm	1	720.5104	3.25	0.0752
Grp*Jm	1	106.2604	0.48	0.4908
Size*Jm	1	906.5104	4.09	0.0465
Grp*Size*Jm	1	2233.0104	10.07	0.0021
Bup	1	168.0104	0.76	0.3866
Grp*Bup	1	27.0938	0.12	0.7276
Size*Bup	1	412.5104	1.86	0.1764
Grp*Size*Bup	1	1113.8438	5.02	0.0278
Jm*Bup	1	1989.2604	8.97	0.0036
Grp*Jm*Bup	1	341.2604	1.54	0.2184
Size*Jm*Bup	1	44.0104	0.20	0.6571
Grp*Size*Jm*Bup	1	8.7604	0.04	0.8429

ANALYSIS OF VARIANCE RESULTS FOR THE NUMBER OF COMMANDS USED DURING PROBLEM SOLUTION TIME (TOTC)
GROUP BY COMMAND SET SIZE BY JUMP BY BACKUP—MENU MODE ONLY

TABLE 28

SAS GRP=1

TOTC		JM 1	JM 2
SIZE			
1	Mean	65.17	77.58
	Std	8.63	15.62
2	Mean	99.25	80.08
	Std	23.56	8.97

SAS GRP=2

TOTC		JM 1	JM 2
SIZE			
1	Mean	100.75	89.67
	Std	20.91	10.53
2	Mean	97.00	92.92
	Std	17.72	13.02

MEAN AND STANDARD DEVIATION FIGURES FOR TOTC
GROUPS BY JUMP BY COMMAND SET SIZE—MENU MODE ONLY

TABLE 29

SAS GRP=1

TOTC		Size 1	2
BUP			
1	Mean	75.00	82.33
	Std	17.37	9.82
2	Mean	67.75	97.00
	Std	8.50	24.94

SAS GRP=2

TOTC		Size 1	2
BUP			
1	Mean	93.08	95.50
	Std	13.67	15.05
2	Mean	97.33	94.42
	Std	20.46	16.30

MEAN AND STANDARD DEVIATION FIGURES FOR TOTC
GROUPS BY COMMAND SET SIZE BY BACKUP—MENU MODE ONLY

terms of usage of commands between option combination of backup and small command set versus backup and large command set. Using option combinations of restart and command set size, the FI subjects used significantly less commands when the option combination was restart and small command set, relatively they used significantly more commands when the option was restart and large command set. FD subjects did not ditfer significantly between the option combination of restart and small command set versus restart and large command set. Comparing the performance pattern of FI and FD, using Table 29, we can see that FI subjects used less commands when option combination was small command set and restart and relatively more commands when the option combination was large command set and restart. FD subjects showed significant difference in command usage under the option combination of backup and small command set and backup and large command set.

A highly significant two factor interaction involving groups is, group by size, $F(1,80)=9.30$ ($p=.0036$). The results again show FI subjects preference for small command set size and FD subjects preference for large command set. The FI subjects consistently used less commands when using small command set and more when using large command set whereas FD subjects used more commands for small command set and relatively less commands for large command set.

In conclusion, FD subjects used single function command (large command set) more frequently than FI subjects. It is clear that FD subjects prefer highly structured approach, specific function commands, and FI subjects prefer highly flexible approach, in which they will have more context manipulation and control. In other words, they prefer the situations where they can give their own

structure. Small command set and backup was a consistent preference of FI subjects, whereas, FD subjects showed overall preference for restart options, where there is least amount of restructuring and basically no manipulation at all.

RS-MCB-MULTIPLE COMPARISON METHODOLOGY

Multiple Comparison Methodology, designated as RS-MCB is a software package for Ranking, Selection, and Multiple Comparison with the Best recently developed by J.C. Shu (1986). This work is based on his theoretical work (1984b, 1981) which is closely related to the Multiple Decision Theory of Ranking and Selection (Bechofer 1954, Gupta 1956, 1965). The purpose of this procedure is to compare three or more treatments. Since one of the main interests of this project was to find what are the best matches of user/interface, it was feasible to utilize this newly developed technique for these data. The results of analyses revealed that the best (in terms of usage of less time and commands) interface match for FD user is sequential. Whereas for FI subjects, the best match interfaces are Keyword as well as Menu with backup and multiple jump. The non-preferred search mode for FI is sequential and for FD it proved to be the Keyword. These results are consistent with analyses of variance results.

SUMMARY AND CONCLUSIONS

The first experiment provided some important support for the basic hypotheses that people's efficiency in using computer systems depends on user cognitive characteristics and design features of human computer interface. However, several questions were raised

from the results of Experiment I. The first and perhaps the most important concerns the underlying mechanisms for group effects—Why did FI users prefer to use interface features differently than FD users? Another question raised was the extent to which FD users prefer small command set involving more generalized commands and FI prefer large command set consisting of more specialized command set.

Experiment II was designed to test and validate the major implication of Experiment I--Do the cognitive characteristics distinguishing FD and FI users significantly interact with interface design features?

The results of Experiment II clearly demonstrated that subject's usage of interface design features significantly interact with his/her fundamental cognitive characteristics--cognitive style. One of the consistent findings was group by search mode. The results clearly revealed that if the interface being used is compatible with his/her information processing style, then it affects the performance positively. In contrast, if the interface design is incompatible with user 's processing style, then his performance deteriorates. FD subjects consistently showed preference for sequential mode. From within menu modes, they preferred the option combination of restart and single jump. Looking at these results, it is clear that FD subject's preference is for search modes involving the least amount of restructuring. For example, if they want to retrieve a target item from the database and if there is a choice between backup or restart, the FD subjects prefer the restart option indicating their tendency of

being context bound. Similar trend is seen in jump option. FD, in general, prefer single jump relative to multiple jump. These results clearly indicate that FD subject perception of database is more of linear format type.

Another significant finding of Experiment II pertains to field independent subjects. They showed strong preference for search modes types which are flexible without any imposed structure such as Keyword. From within Menu modes, FI preferred options of backup and multiple jump. All these results are consistent with Witkins theory of Psychological differentiation, that is, field dependent subjects show a tendency of clinging to the context and field independent subjects are context free. Therefore, FI prefer tasks involving the least amount of structure.

What is most significant and not revealed in the problem solving literature thus far, is that when FI subjects are presented with a highly structured task, their flexible nature does not switch over and adapt to the new situation. In fact, due to the inconsistency of this task with their cognitive style, their performance deteriorates. These results imply that the FI individuals' processing modes are fixed even though they are flexible but when presented with an incompatible situation, their performance suffers.

Now the answer to the question of why the preference for different interface features. The following answer seems to be plausible. In the Keyword mode, subjects are presented with a relatively unstructured database view. The structure or the organization which users perceive is the one which they themselves impose. This structure, in turn, gives rise to their choices of

specific keyword used while searching for items in the database. This structure is generated by the user. This complex database structure seems to be comfortable for FI subjects who are able to identify search paths to information items embedded in a complex structure.

In contrast, the sequential search mode, preferred by FD subjects, presents highly structured, straight forward database view, i.e., a simple linear ordering. Perhaps the preference exhibited by FD users reflects their preference for less complex database views.

One of the main questions raised in the initial stages of this project--does individual difference dimension affect user's information retrieval performance, seems to have an affirmative answer. These results clearly reveal that Human-Computer Interface design can be optimized to individual differences in cognitive style among system operators. The results repeatedly showed that properly matching interface design characteristics can significantly enhance the task performance of human operator.

APPENDIX A

Virtual Interface System (VIS) Section 1

1. Overall System Description

The program VIS Ver 3.3 is a complete and flexible software system which can be used to simulate a wide variety of computer-database user-interface designs. This system can be used for basic experimentation as well as an aid in the design and implementation of actual interface systems.

Basically, VIS 3.3 provides a system whereby a person (e.g., an experimenter) sits at the center of an information-collection network. From the center of this network, the person can access and communicate with a variety of information sources such as other computers and remote databases.

In particular, the experimenter can communicate with another person, the subject, located at a second computer - the "S-computer." The main, central computer at which the experimenter works is the "E-computer."

Since the experimenter can control the form and content of all information presented to the subject on the S-computer display, and can also interpret and process all information entered by the subject into the S-computer, it is possible for the experimenter to make the S-computer mimic any desired database user-interface. The overall system can best be understood by examining the block diagram shown in Figure 1.

1.1 Software Components

The software for this system consists of four basic parts. These are:

EXP2E - This is the primary control program which resides in the E-computer. This program controls the overall flow of information in the system, records experimental data, and most importantly, provides many convenient features that make it easy and quick to format data in a manner consistent with the user-interface being simulated. This includes the ability to create and properly format menus, lists, error messages, etc.

EXP2E consists of four binary program modules, EXP2E, EXP2FB, EXP2EC, and FILESEARCH. Note that EXP2E (or EXP2E.DEMO) is listed as an Applesoft ("A") file when listed. This is because this module consists of a single Applesoft line which jumps to a binary program segment.

EXP2S - This is the control program that resides in the S-computer. EXP2S handles many communication functions, controls the general display formatting of S-computer

displays, and is used to make the flow of the simulation faster by locally handling many tasks (e.g., presentation of "static" screens).

SUPPORT PROGRAMS: A large number of smaller, support programs have been developed for use with this system. These programs provide such functions as downloading EXP2S from the E-computer into the S-computer. The primary components of this program collection are SSC-SETUP, and PROG TRANS LGO (an exec file), PSEUDO-LDB, PSEUDO-MEM-FILES, PSEUDO DISK, FILESEARCH, SETUP (another exec file) and RUN-TIME-PARMS. The functions of these support programs and files are explained in detail in the Appendices.

LOCAL DATABASE FILES - VIS 3.3 provides access to a large set of ready-made responses that can be sent to the S-computer. Utilizing ready-made response files saves the experimenter from entering, in a time consuming, manual fashion, many replies to the S-computer. Response files are provided for PROBLEMS, ERROR MESSAGES, HELP SCREENS, common short phrases [MEMory) Files], and various common ANSWERS, that are frequently sent to the S-computer. In addition, VIS 3.3 allows the experimenter to access one or more optional, on-line, local databases. The primary on-line database is stored in a 128K "RAM Disk" disk drive emulator to provide ultra-fast data search and retrieval.

1.2 Hardware Requirements

The overall system requires the following hardware:

S-computer: an APPLE II+ computer with at least 48K memory; a Videx 80-column display card (equipped with reverse video option) in slot 3; a software-controlled switch (i.e., the Videx "Softswitch") for switching between 40- and 80-column display screens; an Apple "Super Serial" RS-232 communication can in slot 2. A 5 1/4" Floppy disk drive with controller card in slot 6. This disk drive subsystem must be compatible with Apple DOS 3.3.

E-computer: An APPLE II+ computer with at least 64K memory; a Videx 80-column card (equipped with reverse-video option) in slot 3; A Videx "Softswitch" device for video-screen switching: An Apple "Super Serial" RS-232 communication card in slot 2; at least one disk drive with the disk controller card in slot 6; a Mountain Hardware, Inc., "Thunderclock" in slot 7. To support remote database/computer features, a modem card must be provided in slot 4. A Hayes "Micromodem II" is recommended.

In addition, VIS 3.3 requires a 128K RAM memory card in slot 0. This card must be compatible with Saturn Systems software designed for their 128K card.

1.2.1 S- and E-Computer Interconnection

The S- and E-computers are linked via an RS-232 Serial

Communication line between the SSCs in each computer. The SSCs should each be set to the "terminal" mode by properly configuring the onboard switches and jumper block. The baud rate of the S-computer SSC should be initially set to 300b. (EXP2E software will control the E-computer SSC baud rate as well as subsequent settings of the S-computer SSC.)

2. Starting the System

The following procedure can be used to load and start the system.

1. Put a VIS 3.3 PRE-SETUP disk into drive 1 at the E-computer and "boot" it. The PRE-SETUP disk should contain the following files:

- HELLO
- PSEUDO DISK
- PSEUDO.NRM
- PSEUDO.PARAMS

These files are used to prepare the system so that it can store information on the 128K RAM card in slot 0 of the E-computer. This is done by making patches to the DOS that is installed when the PRE-SETUP disk is booted. Note that the DOS that is installed is NOT the normal DOS 3.3.

The following files may also be included on the PRE-SETUP disk although they are optional:

- EXP2S
- LDB.SEQ
- MAKELDB
- INSTRUCTIONS (text files of user instructions)

2. When the PRE-SETUP disk boots, respond to the prompt by pressing the space-bar to RUN the PSEUDO DISK program.

When the PSEUDO DISK program begins, select option3 (INSTALL PSEUDO DISK). (See the Appendix section on the PSEUDO DISK program for an explanation of what all the options do.)

3. After the PSEUDO DISK has been installed, remove the PRE-SETUP disk and put in the SETUP disk in Drive 1 of the E-computer.

The SETUP disk must contain the following files:

- HELLO
- SETUP
- RUN.TIME.PARMS
- PSEUDO-LDB
- PSEUDO-MEM-FILES
- PSEUDO-LDB
- LDB

- EXP2EB
- EXP2EC
- ALL M(emory) files (e.g., M1, M2, etc.)
- ALL HELP files (e.g., HELP.1, HELP.2, etc.)
- ALL PROBLEM files (e.g., PROB1.1, PROB1.2, etc.)

All of these files are REQUIRED to setup VIS 3.3.

Setup all necessary files by typing:

EXEC SETUP (return)

When everything is setup and ready, the computer will display the message:

SETUP IS COMPLETE. INSERT PROGRAM DISK AND RUN EXP2E

(NOTE: This process takes about 7 minutes. You should see the database being loaded into the pseudo disk during the first part of the process.)

4. Now load EXP2S into the S-computer. Put the disk containing EXP2S (this is probably the PRE-SETUP disk) into the S-computer drive and boot the system (e.g., turn on the power). Then type:

RUN EXP2S

The message "PLEASE WAIT..." should appear on the S-computer display. This means the S-computer is waiting for commands from the E-computer. Setup at the S-computer is complete.

5. Now run either EXP2E. Put the RUN-TIME disk in the E-computer drive 1. This disk must contain the following files:

- HELLO
- EXP2E
- ALL ANSWER files (e.g., ANSWER.1, ANSWER.2, etc.)

Since all available space is needed on a RUN TIME disk for data storage, a new RUN TIME disk copy should be prepared for each subject.

RUN EXP2E (return)

The experiment should begin. Continue by answering the prompts presented on the E-computer display. (Remember that the program EXP2S in the S-computer should be running at this point.)

3. Restarting the System After It Has Been Setup

After the system has initially been set up, it is not necessary to repeat all the steps described above. In particular, it is NOT necessary to reload files into the pseudo disk. Files remain in

the pseudo disk until the E-computer is turned off. When the pseudo disk is active, it functions just like another disk drive. To see if files are still loaded into the pseudo disk, catalog it as follows:

CATALOG S5 (return)

After cataloging remember to go back to the main disk in slot 6. This can be done by typing:

CATALOG S6 (return)

Note that the slot designations ARE NECESSARY.

If files are still present, then it is only necessary to type:

- RUN EXP2E

4. Using the System

This section describes how to actually use the system to run an experiment.

4.1 Using EXP2E at the E-computer.

4.1.1 Initial Parameters

Begin by answering the initialization questions presented on the E-computer screen. The acceptable responses to each prompt are shown with each question. The proper responses depend upon the particular experimental conditions and user-interface to be simulated.

4.1.1.1 Some, All, or No Data

This prompt lets the experimenter determine how much data will be collected as the programs run. The "ALL" option means that all data events and the complete contents of ALL messages and displays presented on the S-computer are recorded. This mode requires a lot of data-storage space on the RUN-TIME disk.

The "SOME" option records only data events (e.g., the commands issued by either the subject of experimenter). This saves a lot of disk storage space and makes the programs run faster. This is the RECOMMENDED operating mode.

The "NO" option inhibits data recording. This is used for pilot testing or other situations in which it is not necessary to keep data from an experimental session.

4.1.1.2 Entering HELP-Message Screens

The program will now ask the experimenter to enter the HELP screen that will be presented when the subject the HELP command at the S-computer. Note that this screen will depend very much upon

what interface is being simulated.

The HELP screen that is entered will be basically the same one that is shown to the subject when help is requested. HOWEVER, since some commands will be common to every user-interface, S-computer will automatically put the following lines at the top of every help screen.

COMMANDS

NEW PROBLEM	-	GET A NEW PROBLEM
PROBLEM	-	SHOW CURRENT PROBLEM
HELP OR ?	-	SHOW LIST OF COMMANDS
DONE	-	INDICATE PROBLEM DONE

Remember, the HELP screen is always presented on the 40-column display, so the lines of the HELP screen should be less than 40-columns. To make it easier to see what the screen will look like, EXP2E draws a vertical line on the screen at column 40. If a HELP screen is manually entered, make sure the lines do not cross this boundary.

HELP screens can be entered automatically or manually:

4.1.1.2.1 Automatic HELP-Screen Entry

Ready-made HELP screens may be prepared in advance and stored on the RUN-TIME disk as text files. To access and use a ready-made HELP file, just enter the filename (followed by return).

NOTE: HELP files are stored on disk using names that begin with "HELP." (e.g., HELP.1, HELP.2, etc.). It is only necessary to enter the LAST part of the HELP filename (e.g., entering "1" will access and load helpfile "HELP.1").

4.1.1.2.2 Manual HELP-Screen Entry

First, just press RETURN to specify manual entry. Then, type in the desired HELP lines (remember to watch to 40-column boundary line). To end entry of the HELP screen, press "*" (no return is needed).

HELP screens may have embedded carriage returns. However, only a MAXIMUM of 17 lines should be entered so the entire HELP screen (along with the standard HELP lines described above) will fit on the 40-column x 24 line display.

HELP-SCREENS MUST BE ENTERED IN UPPER-CASE ONLY. LOWER-CASE LETTERS ARE NOT ALLOWED IN HELP-SCREENS.

4.1.1.2.3 Accepting, Correcting or Re-entering the HELP-Screen

If HELP-screen is OK, press "Y" to next question on screen. If HELP screen is not OK press any other key to re-enter the HELP display. After the HELP screen is accepted, it will automatically

be sent to the S-computer along with all other experiment parameters. The E-computer should display:

LOADING PARAMETERS INTO S-COMPUTER
PLEASE WAIT

When all parameters are loaded, the E-computer will display:

S-computer is Ready
Press any key to continue

At this point all parameters have been loaded into the S-computer and the S-computer should be waiting to begin the experiment. The S-computer should be in the 80-column reverse video-mode. A "?" should be displayed in the upper left corner of the S-computer display.

Also, the E-computer is ready to begin. At this point, make sure the subject is ready. Read the instructions to the subject etc., and when ready, start the experiment process as described below:

4.1.2 Beginning the Actual Experiment Session

When ready to actually start, press any key as instructed on the E-computer display. This will activate the CCM level described below.

4.1.2.1 Communications Channel Menu (CCM)

The CCM is the "top level" of the EXP2E program. From the CCM the experimenter can choose to communicate with a variety of information channels and sources.

Most channels should be self-explanatory and so only a short description of each channel's function will be given.

4.1.2.1.1 DOS 3.3

"DOS 3.3" is the path out from VIS 3.3 and should only be used at the END of the experiment session. (Note that VIS 3.3 actually uses DAVID DOS - a modified version of DOS 3.3.)

4.1.2.1.2 REMOTE DATABASE

This option allows the experimenter to communicate with a remote database connected to the E-computer via a modem. The remote database connection must be established BEFORE running EXP2E. The link must be functioning before attempting to use this process or the program will hang up, waiting for a reply from the remote computer. The messages sent and received do not affect any interactions between the S- and E-computers.

4.1.2.1.3 University Computers

This option is intended for use when some local university computer is connected, via the modem, to the E-computer. This option is basically similar to the "Remote Database" option but is intended to support more extensive interactions, such as transferring data from the E-computer for analysis. Note that this option and the "Remote Database" option cannot be used at the same time since only one modem is available on the E-computer.

4.1.2.1.4 Experimenter's Program

This option is designed to allow a special temporary exit from EXP2E to any special subroutines in the E-computer which may be written for customized future versions of VIS 3.3. Re-entry to EXP2E from such a subroutine should be via a jump to location $1100 where the code for EXP2E begins.

4.1.2.1.5 Subject's Console

Normally, the experimenter will wish to communicate with the S-computer. This can be accomplished by choosing channel "s".

4.1.2.2 The SCOMP Menu Level

Selecting channel S moves the experimenter to the "SCOMP" communications level. At this level, there are several options for communicating with the S-computer. These options are explained below:

4.1.2.2.1 Other Communications With Subject

This option allows the experimenter to send a single message to the S-computer. The person at the S-computer then can send a single reply. This option is used mainly for testing the communication between the S- and E-computers. This communication option is outside the flow of the main experiment-control process and so should not be used when an experimental session is in progress.

Note: Messages sent back and forth must be LESS THAN 256 characters long.

THESE MESSAGES MUST BE ENTERED IN UPPER-CASE ONLY. LOWER-CASE LETTERS ARE NOT ALLOWED SINCE THEY WILL BE DISPLAYED ON THE S-COMPUTER'S 40 COLUMN SCREEN.

4.1.2.2.2 Local A

This option is reversed for future versions of the program and have no effect in the current Ver 3.3.

4.1.2.2.3 Return to Main Communications Console

This option allows the experimenter to go back to the CCM level. For example, during the course of the experiment, the

experimenter might wish to access the remote database. This can be done by selecting option 6, then selecting the Remote Database option on the CCM menu, and then finally returning from the CCM menu to the SCOMP level.

4.1.2.2.4 Leaving the Program

This option allows the experimenter to leave the program when the experiment session is over. To exit from the system, go back to the CCM level and then exit by choosing the DOS 3.3 option.

4.1.2.2.5 Begin Experiment Session

This option should be chosen to start an experiment session. The main purpose of this option is to initialize the datafiles used in the experiment.

4.1.2.2.6 Send New Problem to Subject

This option can be selected to send a new problem to the subject. As with the HELP-screen, problems can be entered either manually or automatically.

4.1.2.2.6.1 Automatic Problem Entry

Ready-made files may be stored as text files on disk 1. These files all have names that begin with "PROB1.". When requesting one of these problem files, it is not necessary to enter the first part of the problem file name. For example, to access problem "PROB1.1", just enter "1" (followed by return).

4.1.2.2.6.2 Manual Problem Entry

Problems may also be entered manually. Remember that problems are presented on the 40-column screen. Therefore problem lines should be less than 40 columns wide.

To make it easier to properly enter a problem, a vertical line is drawn on the display at column 40. Do not go over this boundary when manually entering a problem.

When entering a problem, you can use the backspace key to correct typing errors.

To terminate manual problem entry press "*" (no return is needed).

NOTE: The TOTAL number of characters in a PROBLEM screen must be LESS THAN 256.

USE ONLY UPPER-CASE WHEN ENTERING PROBLEMS. LOWER-CASE LETTERS ARE NOT ALLOWED.

4.1.2.2.6.3 Accepting, Correcting, or Re-entering a Problem

After a problem has been entered, type "Y" to accept it and send it to the S-computer. Pressing any other key will restart the problem-entry process.

4.1.2.2.6.4 Continuing After Sending a New Problem

Once the problem is received at the S-computer, it will automatically appear on the S-computer 40-column screen. The E-computer will wait until the subject has read the problem and pressed a key indicating that s/he is ready to continue. When this key-press is made, the main flow of the program will automatically continue.

After the new problem has been sent, the program will return to the SCOMP menu. The experimenter should now select option 7 (CR LOOP) and proceed with the experiment.

4.1.2.2.7 Command/Response Sequence

This option causes the program to enter the Command/Response Sequence Loop. This is the heart of the experiment process. It is in this loop that the subject enters commands to the "virtual database," the experimenter receives these commands, processes them, and then enters a response which is sent back to the S-computer. This back and forth command-response loop continues until the experimenter decides to terminate the process.

The experimenter may terminate the process for several reasons. For example, (and most commonly), the subject finishes (i.e., solves) a problem and the experimenter terminates the CR loop so that a new problem can be sent to the subject.

Thus, in the CR loop, the experimenter must receive and monitor commands entered by the subject and must then enter, format and send responses back. The program provides many features to make this task easier and faster.

When the CR loop begins, the E-computer waits for the subject to enter a command. (Remember, the subject has already received a problem.)

4.1.2.2.7.1 Responding to Commands from S-Computer

When a command is received from the S-computer it is displayed on the experimenter's screen. The experimenter must now decide how to respond to the command.

The response to be made will depend upon the type of interface being simulated, as well as the specific command. Several options are available for responding, as described below:

-IGNORE: The experimenter may wish to "ignore" the command and simply wait for the next command without doing anything.

For example, if the subject has requested to see the "help" screen by entering the "HELP" command, there is no need for the experimenter to do anything - the "help" screen is automatically presented by the S-computer. Thus the experimenter would simply "ignore" this command and continue to wait for the next command.

NOTE: Even though the experimenter "ignores" a command, all pertinent data concerning the command is still automatically recorded. So "ignoring" a command does NOT mean that data concerning this command is lost.

Another command that should be ignored is "PROBLEM", since this is also handled automatically by the S-computer.

USE "IGNORE" ONLY FOR "HELP" AND "PROBLEM" COMMANDS. Using IGNORE with any other command will cause the system to hang-up.

-LEAVE CRLOOP; RETURN TO SCOMP MENU: The experimenter may terminate the CRLOOP interaction directly at this point and return to the SCOMP menu. This would be done, for example, if the subject indicated that they had "solved" the problem upon which they were working. The experimenter could then go to the SCOMP level to send a new problem or to end the experimental session. To leave the CRLOOP, enter CONTROL-X (followed by return). NOTE: This exit option is NOT shown in the choice-menu at the bottom of the screen - however, it is available.

-SELECT A READY-MADE RESPONSE FROM DISK: Depending upon the interface being simulated, the experimenter may have stored some common responses as text files on disk 1. Each such text file is stored on disk 1 with a name beginning with "ANSWER.". To access a disk answer file, only the second half of the name need be entered. For example, to access "ANSWER.1", the experimenter just enters "1" (followed by return).

-SELECT A READY-MADE RESPONSE FROM MEMORY: As an alternative to retrieving ready-made responses from a disk, the experimenter may retrieve ready-made responses which have been stored ahead of time in memory. This is advantageous since retrieving responses from memory is much faster than retrieving them from disk. It is recommended that as many short, common responses as possible be stored in memory for easy access. The program allows storage of 50 such ready-made answers.

To use a ready made answer from memory, enter ".nn" where "nn" represents the code number of the desired memory file. For example, to use memory file M1, enter ".1 (return)".

Ready-made responses are loaded from the RAM DISK when the program starts. Memory files are permanently stored on the SETUP DISK as text files with filenames of the form: M1, M2,

3- /,99/

Note, however, that only ONE ANSWER file can be specified in a single command.

*** Automatic Lines from On-Line Database. Items are stored as separate records in an optional random-access database file on the 128K RAM card "pseudo disk." Single records or a range of records may be accessed and automatically added to the manual response-line list. To specify retrieval of records from the database, use the following command format (NOTE: commands MUST be terminated with a "/"):

1- /18,56,17,25-30,109/

Where "1-" is the line number, not part of the command.

The command is interpreted as follows: First get item 18, then item, 56, then 17, then retrieve the RANGE of records 25-30 (i.e., retrieve records 25, 26, 27, 28, 29, and 30), then finally retrieve record 109. Each record is presented on a separate line. Thus, with one command, the experimenter has generated a list of TEN items. Notice that each item or range of items must be separated by commas.

*** Retrieving Ready-Made Line from Memory. Any of the previously stored memory items may be automatically added to a line. For example to use memory item 1 on line 5, the experimenter would enter:

5- /.1/

Notice that memory items are indicated with the ".". (Remember that ANSWER files are indicated by a ",".) Only ONE memory item may be specified in a command line.

*** Retrieving from Other Database Files. It is possible to change the database file from which records are retrieved. The new database MUST be a random-access text file. The name of the file MUST be three characters in length. The normal database from which items are retrieved is named LDB (local database). To retrieve from an alternate database file, us the following command format:

6- /XYZsdll,15,20-25/

This changes the database filename to XYZ. The slot of the drive containing the database disk is specified by "s" (typically 5 or 6). The drive is specified by "d" (either 1 or 2), and the length of records in the random-access file is specified by "ll". Notice, "ll" MUST be a two-digit number. The comma following the record length IS REQUIRED. ALL 8 CHARACTERS MUST be present when changing the database file. After the file is changed, the program will retrieve record 15, and records 20-25.

The change in database file REMAINS UNTIL IT IS CHANGED AGAIN. The normal file is specified by /LDB5143, (i.e., the name is LDB, it is on "pseudo drive 1", and has a record length of 43).

*** Using Data Obtained from Local Database Search. Information retrieved from the local database can be automatically added to the response list by using the "U" option (i.e., USE). After retrieving information from the local database and returning to the manual response entry level, select U option as follows:

/U/

The retrieved data will automatically be added to the response list.

After a response has been entered, press "*". Note that the response lines are NOT final. They can be changed later using the EDITOR process which is invoked when the "*" key is pressed.

4.1.2.2.7.3 Searching the Local Database

In generating a response to the subject, it may be useful to search the local database. A high-speed, keyword database is built into EXP2E. To access the local database (LDB) enter CONTROL-E from the manual line entry process (BEFORE pressing "*" as mentioned above). The LDB display will appear. Two search options are available:

FIND - This mode scans the ENTIRE database and finds ALL matching items. To search for all items containing the word "PORK", enter:

FPORK (return)

SEARCH - This mode finds each matching item one at a time. The experimenter can either continue or stop the search after each match. To use this option enter for example:

SPORK (return)

A "wildcard" character "?" is also available. As an example, entering FP??K would find all matches to PORK and also find all matches to PACK, PEEK, etc.

Also, the matching process finds partial matches. Thus "SPO" would search for and match PORK as well as POTATO.

ONLY ONE keyword may be specified in any search (e.g., APPLE PIE is NOT allowed. Separate search for APPLE and for PIE would need to be made).

After a search is made with one keyword, another search can be made with a new keyword. Three options are provided:

NEW - With this option previously-retrieved items are NOT retained for transfer back to the response-line list after the next search with the new keyword.

APPEND - New retrieved items are ADDED to the list of all previously retrieved items. Thus, one can retrieve all items containing the keyword "PORK" and then add to this list all items containing the word "BEEF".

RETURN - Return to the manual response entry display.

Note that retrieved items can be added to the response-line list with the /U/ command (see section 4.1.2.2.7.2).

4.1.2.2.7.4 Leaving the CR LOOP

The experimenter can leave the CR Loop permanently or temporarily. Leave the CR Loop PERMANENTLY when a problem has been completed, for example. Leave the CR Loop temporarily to access information from a remote database and then return to make the response to the subject, for example.

To leave PERMANENTLY, press cntrl-X.

To leave temporarily, press cntrl-W. To return after temporarily leaving, just select option 7, Command/Response Sequence, from the SCOMP communications menu. The program will automatically return you to just where you left.

4.1.2.2.7.5 Editing Response Lines

After a basic response has been entered, it may contain errors or the experimenter might want to review it. This can be done with the EDITing function. After the experimenter types "*" to complete the basic entry of the problem, the EDIT function is automatically started. The following commands are available:

```
+(return) ..... show the next "page" of 11 items
-(return) ..... show the previous page
A(return) ..... add a new line to END of response list
Dn(return) ..... delete line n
Rn(return) ..... replace line n
In(return) ..... insert a new line in front of line n
```

When the response list is as desired, just press RETURN (with no command) to leave the editor.

4.1.2.2.7.6 Automatic Formatting

The system will automatically format a response in several ways. For example, the system can automatically produce MENUs or LISTs on the subject's display. To do this, put "formatting"

commands into the response lines. This can be done initially or with the EDIT function. The following format commands are available:

```
.M ............. start of end a menu
.L ............. start or end a list
.LMnn .......... set left margin to nn
.LM ............ end a current margin setting
.C ............. start or end automatic line centering
```

The use of these commands is best shown by examples:

These response lines:

```
1- .m
2- apples
3- pears
4- plums
5- .m
6-
7- .c
8- choose from above
9- .c
10- bananas
```

produce:

```
*** Menu of Choices ***
     1- apples
     2- plums
     3- pears

choose from above
```

bananas

Notice that the final item is not centered, because the second occurrence of a formatting command (i.e., the second ".c" in line 9) disables the centering function.

A list is basically similar to a menu but the heading "*** Menu of Choices ***" is not used.

Changing left margin is illustrated in the following list:

```
1- .lm10
2- apples
3- .lm20
4- plums
5- .lm
6- pears
```

produces:

apples
plums
pears

The margin setting function can be used to easily center all response items on the S-computer display without having to type in a lot of spaces.

When the Automatic Formatter has completed its operation, if any changes to the display were made, the EDIT function is again automatically activated. This allows the experimenter to review the formatted response and make any corrections if needed. New formatting instructions can be given if necessary.

To send the completed, formatted response to the S-computer, just press RETURN from the EDIT function choice line.

This completes the description of EXP2E operation.

4.2 Operations at the S-computer.

Unlike operations at the E-computer, nearly everything that happens at the S-computer should be self-explanatory.

The only significant point to note is that a DEMO version of EXP2S can be created for testing purposes by changing the definition of SSC% 2 to 0. (approximately line 200 in the program).

5. Data Collection

All responses made by the experimenter or subject are automatically recorded and stored on the disk in drive 1 of the E-computer. The data file is named "DATA.SUBJNAME" where SUBJNAME is the name of the subject.

In addition, all displays presented to the subject are completely recorded, line by line.

The times of all responses (either by E or by S) are recorded to the nearest 1 second.

5.1 Data Collection Options

Recording the complete data from an experimental session requires a large amount of disk space. For testing or other reasons, it may be useful to record only partial data. Two options are provided for this. These options are selected when the program is started.

Option N (No Data) can be selected to eliminate any data recording. This is useful primarily for testing purposes.

Option S (Some Data) collects all event data but does not dump out a complete copy of each screen which is sent to the S-computer. This option is probably the best way to collect data in normal situations.

APPENDIX A--Section 5
Text File Formats

EXP2E uses numerous text files which are prepared in advance for presentation during an experiment session. All these files are standard text format and are most easily prepared with APPLEWRITER II or some similar text editor. The formats of these files are as follows:

MEMORY FILES. These are short answers that are commonly needed as responses to the subject (e.g., "No Matching Items," "Incorrect Command," etc.) Filename format is Mn where n is an access code number. For example, M1, M2,,,,M10, M11, etc. Memory files must be less than 256 characters long and must end with "*". Memory files may contain multiple lines (i.e., carriage returns). For example:

No matching Items*

ANSWER FILES. These are text files of ready made answers that can be sent to the S-computer. They are intended to be used when the subject requests details of a particular recipe. Answer files may contain embedded carriage returns and may be MORE than 256 characters long. However, each ANSWER FILE MUST end with "*". ANSWER filenames have the following format: ANSWER.word. "word" may be any word or number (e.g., ANSWER.1, ANSWER.PORK, ANSWER.VEGETABLES). For example:

1500 Calories
Serves 5 People
Takes 3 Hours to Prepare*

PROBLEM FILES. Problem Files are text files that contain ready made problems to be presented to subjects. Problem filenames have the following format: PROB1.word, where "word" may be any word or number (e.g., PROB1.1, PROB1.SALAD, PROB1.BEEF, PROB1.CHINESE). Problem files must be LESS THAN 256 characters but they may contain embedded carriage returns. Problem files must be designed for display on a 40-COL SCREEN.

Problem files MUST USE ONLY UPPER-CASE LETTERS and MUST end with "*". For example:

FIND A RECIPE FOR A CHINESE ENTREE THAT HAS
LESS THAN 120 CALORIES PER SERVING*

HELP FILES. Help files are text files which provide HELP instructions appropriate for the particular interface being simulated. Help files must be designed for display on a 40-COL SCREEN. Help filenames have the format: HELP.nn where "nn" is an access code number (e.g., HELP.1, HELP.2). The proper format for a Help display can best be seen by looking at the default Help display on the S-computer.

HELP FILES MUST BE ALL UPPER-CASE and MUST end with "*". For example:

```
SEARCH ABCD  -  SEARCH FOR WORD "ABCD"
BACKUP       -  SHOW PREVIOUS MENU*
```

NOTE: Help files may contain embedded carriage returns. Also, Help files can be LONGER than 256 characters (unlike some other text files). However, Help files MUST contain a MAXIMUM of 18 lines since additional lines will be automatically pre-pended. If more than 18 lines are used, then the entire Help display will not fit on the 40-column screen.

APPENDIX A—Section 6
Changing the RUN-TIME Parameters

EXP2E Ver 3.3 uses a special text file called RUN-TIME-PARMS to control the run time environment. This allows EXP2E Ver 3.3 to automatically set several parameters which were formerly entered manually (e.g., SYNC DELAY). These parameters can be altered by changing the RUN-TIME-PARMS text file. This is simply a text file containing a list of parameters. The order of parameters is:

SYNC DELAY
Low Baud Rate Code
High Baud Rate Code
Number of Memory Files to be Loaded
Record Length of LDB File
Number of Problem Files to be Loaded
Number of Help Files to be Loaded
Slot Number for the Printer Interface
Slot Number for Super Serial Card
Slot Number for the Videx Card
Slot Number for the Modem Card
Slot Number for the Clock Card

Any of these parameters may be changed with a text editor such as APPLEWRITER II.

The standard SYNC DELAY value is 150. This is a delay value used to synchronize the S- and E-computers during data transfers.

The low and high baud rate codes control the rate at which the Super Serial Cards in each computer communicate with each other. The following table of codes is applicable:

4 ..... 75 baud
5 .....150 baud
6 .....300 baud
7 .....600 baud
8 .....1200 baud

Standard values are 5 and 5 (i.e., 150 baud is used for both high and low speed operations).

The other options are basically self-explanatory. However, the final option - Slot for Serial Communication - requires more detail. This option is used to produce a "demo" version of the EXP2E program as explained in Appendix F.

APPENDIX A—Section 7
Creating a Demo Version of the EXP2E Program

During development and for testing purposes it is often convenient to use a "demo" version of the VIS 3.3 system that can be run with only one computer - the E-computer. This facilitates testing a practice because a second person is not needed at the S-computer. Instead, all information that would be entered by the Subject is actually entered at the E-computer keyboard.

This also means that the experimenter must manually generate all the acknowledgment signals that would normally be produced automatically by the S-computer program.

To create a "demo" program change the Super Serial Card Slot option in the RUN.TIME.PARMS file to "0." For normal operation, this option should be set to "2."

APPENDIX A—Section 8
Functions of Other Program Modules

The functions of various other software modules in the VIS system are as follows:

EXP2E.DEMO This is a version of the basic EXP2E module that has been modified by directing all Super Serial Card I/O to the E-computer keyboard and 40-column screen instead. By using the EXP2E.DEMO, the experimenter can test the system without loading EXP2S into the S-computer. All responses that would normally be coming from the S-computer must be entered at the E-computer keyboard. Responses received from the "S-computer" are not normally visible (since they are displayed on the 40-column screen only).

EXP2EB This is the second binary overlay module. This segment primarily controls the CCM and SCOMP menu functions. The entry point for this module is $1100.

EXP2EC This is the third binary overlay module. It is primarily concerned with processing during the CRLOOP. It therefore handles functions such as automatic formatting, editing, and linking to the database search functions. The starting address is $1100.

FILESEARCH This is a short binary module which resides in upper memory just below the DOS file buffers. This segment is used for database searching and performs all matching operations. This segment's starting address is $9100.

PSEUDO-LDB This is an Applesoft program which transfers the local database text file, LDB, from the setup disk to the pseudo disk.

PSEUDO-MEM-FILES This is an Applesoft program which transfers the various memory files from the setup disk to the pseudo disk. This program also transfers the RUN.TIME.PARMS file to the pseudo disk.

APPENDIX A—Section 9
Updating and Correcting the Local Database, LDB

The Local Database, LDB, is stored as a random-access text file. EXP2E expects this file to contain 5 fields in each record. The first field contains item level codes (e.g., 01, 02, 03, etc.). Fields 2-4 contain the actual item content of the record (e.g., "pork chops") while field 5 contains page information (e.g., 123-124).

This LDB file is created from the basic, sequential text file version of the database. The sequential text file version is generated by APPLEWRITER II or some similar text editor.

Therefore, to update LDB, first make any corrections to the original, sequential version text file version of the database. The name of this sequential file should be "LDB.SEQ."

Next put a blank, formatted disk into Drive 2. Load the program MAKELDB which is on the PRE-SETUP disk. Then put the disk containing LDB.SEQ into Drive 1 and run the MAKELDB program (i.e., type "RUN MAKELDB <return>").

The program will ask for the number of records in the LDB.SEQ file. This number must be entered correctly. In case this number is not known for sure, MAKELDB has a feature that will automatically count the number of records in the LDB.SEQ file. However, this process takes several minutes and can be skipped if the number of records is known in advance.

After the number of records in the LDB.SEQ file has been entered the MAKELDB program will ask for the record length to be used in the new random-access LDB file.

The standard record length is 40. Note that this includes space for page number and level code information. Thirty character spaces are left for item information. Recipe information which exceeds 30 characters will be truncated.

The new LDB can be made more compact by using a record length less than 40. The number of characters allocated for item information is (record length)-13.

If a value other than 40 is entered, it will be necessary to also change the RUN.TIME.PARMS file to reflect the new record length.

After the necessary values are entered, the program will run automatically. The program requires about 15 minutes to create the new LDB file.

APPENDIX B

Description of Simulated Interfaces

Twelve simulated user-interfaces have been developed for testing the Virtual Interface System (V.I.S.). For each simulated interface, a set of commands has been devised and these have been arranged in HELP files stored on the SETUP disk. The reader should examine these HELP files when going through the following explanations of each simulated interface.

The interfaces have been designed to include the major experimental factors in the "Individual Differences in Human-Computer Interaction" Project. Briefly, these are:

1. Size of command set—large/small
2. Search mode—keyword or menu
3. Backup versus restart only
4. Jump mode—multiple versus single-jump

These factors have been combined to yield the various experimental interfaces. An explanation of each interface now follows:

Interface 1. Key Word (KW), Small Command Set

This interface uses KW search mode and only one general purpose search command "SEARCH." By adding various options to the "search" command, different "search type" operations are performed. The basic concept of this interface is that when the user makes a search, a new, smaller list of "matched" items of "hits" is generated. This is called the "search list."

When the user starts, the Search List is the entire database, since no matches have yet occurred. When the user makes a search, the resulting list of matched items will be the next list which is searched.

For example, the user may first search for matches to the keyword "PIE." All items containing this keyword are retrieved and make up a new list. If the user then searches for matches to the keyword "APPLE," only the "search list" will be examined. Thus, the user will probably match "apple pie" but not "apple cider" or "apple butter" because these items would not be in the Search List after the first search.

Thus, the following Search List rule applies. After a search, ANY matching items cause the old Search List to be ERASED and replaced with the newly-matched items. However, if NO matches occur, then the old Search List is unchanged (i.e., is NOT erased).

The user can reset the Search List and cause the system to search through the entire database, rather than the current, limited search list.

The other important concept is "SEARCH CATEGORY." The user may specify that searches are to be made only on items within certain "categories." Note that categories do NOT have to be part of the actual name of any item in the database. For example, suppose the user wants to search only for "desserts" made with apples. The appropriate "Search Category" might be "desserts." However, note that the word "desserts" is not part of the name of any item in the database. Thus, if the "Search List" method was the only way that searches could be restricted to less than the entire database, then it would not be possible, for example, to only search for "desserts." By specifying a "Search Category" the user can limit subsequent database searches only to items in the names category. The category can be changed at any time. It can also be "cleared" so that the entire database is searched without any category restriction.

Finally, a distinction is drawn between matching/retrieving items and displaying them. In this interface, the user first matches items. When an item is matched it is automatically "retrieved" (i.e., made available for display on the user's screen). However, it is not actually displayed until the user requests it to be. The user is normally only told how many items are matched as the outcome of a search. Thus, if the user accidentally enters a keyword that matches hundreds of items, he/she is not forced to watch all these items as they are displayed on the screen. Instead, the user would be told that "too many" items were matched and would then enter another keyword to pull out a smaller set of items from the large, initial Search List.

For example, if the user searches for all matches to "apple" perhaps 200 items will be retrieved. The user is informed that 200 matches resulted from the search. The user then searches for "pie." Now, only items containing BOTH the words "apple" and "pie" are retrieved. This may be only 10 items. In this case the user may request that all items in the resulting 10-item Search List be displayed. A command option is provided for displaying the Search List.

After the user sees the item desired in the displayed Search List, he may do another search using exactly the name of the desired item. Thus, if the user sees "French Apple Pie" in the displayed Search List he may enter this name exactly. Then one and only one item will be matched. The resulting "final Search List" will have only one item in it--French Apple Pie.

Since the desired item has been retrieved, the user may now wish to see whether the choice is appropriate. When (and only when) exactly one item is matched can the user request to see details for "Recipe Choice."

This description explains the basic structure of the interface. The commands should be more or less self-explanatory when considered in light of this structure. For example, there is a command for defining a Search Category. To set the Search Category

to "desserts" the user would enter: SEARCH "desserts

A "wildcard" character, &, is provided which automatically "matches" any item. This character is used just the same as a normal keyword. For example, if the user entered: SEARCH & then ALL items in the current search list would be counted as "matches." This "wildcard" property is used in combination with other commands to produce new results. For example, to display all the items in the current Search List the user would enter: SEARCH &,d

Similarly, if the user wishes to see the number of items in the current search list (for example, to determine if the Search List is small enough to display), then the following command would be used: SEARCH &,#

Even if current Search List or Search Category restrictions are in effect, the user may still search through the entire database without restrictions. This will not affect the current restrictions: these will still be in effect for later searches. To override any current restrictions and make an "unrestricted" search enter the following: SEARCH Kw,&

The reader will notice that all these commands are options to the basic command "SEARCH." Thus, the basic keyword search command is: SEARCH Kw where "Kw" is the term the user wishes to match. Some other examples are as follows:

To search and automatically display all matches (if the number is less than the value set as "maximum" by the experimenter), the user would enter: SEARCH Kw,D

To "reset" the Search List, the user enters: SEARCH i.e., NO keyword is entered. This causes the current, limited Search List to be reset to the beginning value—the entire database. Then the next search will be made on the entire database, rather than just the current Search List. However, if any matches are made, a new Search List will be created.

To similarly reset the Search Category, use: SEARCH " i.e., NO category name is given. Thus, the Search Category name is "cleared" or "reset" so that searches will be made of the entire database until a new Search Category is defined.

If the user forgets the current Search Category name, the system displays the name when the following command is given: SEARCH "?

To search and automatically display the recipe if exactly one match is found, the user enters the command and options shown: SEARCH Kw,R

The remaining commands can be understood from HELP file for this interface:

```
SEARCH (OPTION); WHERE OPTION IS:

KW.......SEARCH FOR "KW"
"BLANK"..RESET SEARCH LIST
KW,D.....SEARCH FOR KW; SHOW MATCHES
KW,R.....MATCH KW; SHOW RECIPE IF ONLY 1 MATCH
KW,&.....SEARCH, UNRESTRICTED
KW,#.....SEARCH; SHOW # OF MATCHES
"NAME....SET SEARCH CAT.="NAME
&,D......SHOW ALL ITEMS IN LIST
&,#......SHOW SEARCH LIST SIZE
RECIPE
CHOICE..SHOWS THE MESSAGE "GOOD CHOICE"
```

Interface 2. KW—Large Command Set

Interface 2 is similar to Interface 1. The basic structure is identical. In both interfaces, Search Lists and Search Categories are used. Also, both Interfaces use KW Search Mode. The only major difference is that Interface 2 uses a large number of special-purpose commands to perform the operations that were specified by adding options to the SEARCH command in Interface 1.

Refer to the explanation of Interface for information on the concepts of Search Lists and Search Categories. Also, for information on the difference between matching/retrieving and displaying items. Retrieved items are only displayed if the user explicitly requests this to be done.

Examples of how to use the various special purpose commands will now be given:

To perform a basic search, use the command SEARCH: SEARCH Kw where "Kw" is the keyword or phrase to be searched for. The rules for matching keywords are the same as those of Interface 1.

To reset the category (i.e., make the Search Category) so that the entire database, rather than just a limited category will be searched, use: CAT (Recall that with Interface 1, this would be done with the command option: SEARCH "). To reset the Search List to equal the entire database, use: LIST

After a Search List has been retrieved, the user can request to display it as follows: DISPLAY

Even if a Search Category or Search List is in effect, the user can cause the entire database (rather than just the current Search Category and Search List) to be searched by using the command "MATCH" rather than SEARCH. For example: MATCH Kw

This command would look for all matches to "Kw" anywhere in the database, not just the current category or Search List. Thus, this command is equivalent to: SEARCH Kw,& in Interface 1.

To define a Search Category, the following command is used: MAKE CAT name where "name" is the name of the new Search Category.

The user continues to make selections from the current Search List, making the Search List smaller and smaller. When only one item remains in the search list, then the user can see the recipe associated with that item by entering the command: RECIPE

Note that if the user makes a search, the newly matched items constitute the new Search List. If a search results in NO matches, however, then the previous Search List is left UNCHANGED, i.e., the old Search List is NOT erased if no matches are made. But if ANY matches ARE made, the old Search List is ERASED and REPLACED BY THE NEW MATCHES.

The main commands of Interface 2 are shown in its HELP file:

```
SEARCH KW.......SEARCH FOR "KW"
DETAIL KW.......MATCH KW; SHOW RECIPE IF ONLY 1 MATCH
CAT.............RESET CATEGORY NAME
LIST............RESET SEARCH LIST
DISPLAY.........SHOW SEARCH LIST
COUNT KW........SEARCH FOR KW; SHOW # OF MATCHES
MAKE CAT NAME...SET SEARCH CAT.=NAME
SIZE............SHOW SEARCH LIST SIZE
SHOW CAT........SHOW CURRENT CAT. NAME
RECIPE..........SHOW RECIPE OF LONE SEARCH LIST ITEM
MATCH...........SEARCH, UNRESTRICTED*
```

Interface 3. SEQ, Small Command Set

The structure of Interface 3 is a simple numerical sequence. To the user, the database simply seems to be an ordered list of recipe names, alphabetically arranged. Thus, the user's database view is a linear sequence. The commands provided in this database allow the user to retrieve one or more items by specifying their sequence numbers. A range of items (e.g., 17-34) can also be retrieved.

The only other basic concept is that of a PAGE of items. This means 15 or so items that are grouped together. For example if items 15-30 are currently being displayed, the user can request the next Page of items, (e.g., 31-45) or the previous Page (1-15).

Interface 3 uses only a "Small Command Set" so all the commands are basically options done with the single command: DISPLAY. The various operations that can be done with the DISPLAY command are as follows:

To display a single item (for example item number 27) enter: DISPLAY 27

To display a range of items (say, items 27-35) enter: DISPLAY 27-35

The user could also specify both a single item (or items) as well as a range (or ranges). For example, to retrieve and display item 3, 6, 15-20, 27, and 30-35, the user could enter: DISPLAY 3,6,15-20,27,30-35

Note that this is very similar to the format that is used by the experimenter in specifying which items should be retrieved from the LDB when in the CRLOOP of EXP2E. The experimenter must watch to make sure the item numbers are entered properly and do not specify too large a range. For example, the following commands should result in errors: DISPLAY 29880 DISPLAY 35-27 DISPLAY 1-500

If the top-most item currently being displayed is number X (e.g., if items 25-32 are being displayed, then X=25) then the user can move back N items (for example 23 items) by the following command: DISPLAY -23

Since 7 items were on the screen, now, after moving back 23 items, the screen should show 7 new items, 2-9 (i.e., the new display begins with 25-23=2 and shows 7 items). Similarly, the user could jump ahead from the bottom-most number on the display screen. For example, if items 25-32 were being shown, the user could jump ahead 5 items by entering: DISPLAY +5

This would cause items 30-37 to be on the display screen. The experimenter must make sure that the user does not enter a command that would go past either end of the database sequence. For example, if items 3-10 were being displayed, the following commands should be errors: DISPLAY -15 DISPLAY +9997

To show the recipe of a single item, the user enters, for example: DISPLAY 15,R to display the recipe of item 15.

To move ahead or back 1 page from the currently displayed items, the user could enter "DISPLAY +8 or DISPLAY -10" for example. However, to automatically move ahead or back 12 items, use the "Page" option as follows: DISPLAY -P or DISPLAY +P

Thus, if items 20-30 were on the display, then "DISPLAY -P" would cause items 8-20 to be displayed; "DISPLAY +P" would cause items 30-32 to be displayed. Note that a "full page" of items is always displayed: Even though only 10 items were being shown, when the :display next/previous page" option is used, the next or previous FULL page of items is shown. If this goes beyond the limit of the database, then the message "No More Pages" or equivalent should be given. (For example, if items 1-12 are being displayed, DISPLAY -P should result in the "No More Pages" response; if items 3-10 are being displayed, the DISPLAY -P should result in items 1-12 being displayed.)

The user may also specify a starting point at which the paging operation should begin. For example, to show the 12-item page which BEGINS with item 27 (i.e., items 27-39) the user would enter: DISPLAY 27,P

But to display the page ENDING with item 27 (i.e., items 15-27) the user would enter: DISPLAY P,27

The HELP file for Interface 3 is shown below:

```
DISPLAY (OPTION); WHERE OPTION IS:

N1-N2.....SHOW ITEMS IN RANGE N1-N2
N1.........SHOW ITEM N1
-N1........MOVE BACK N1 ITEMS
+N1........MOVE AHEAD N1 ITEMS
N1,N2-N3...SHOW ITEMS N1 & N2-N3
N1,R.......GET ITEM N1 & SHOW RECIPE
-P.........DISPLAY BACK 1 PAGE
+P.........DISPLAY FORWARD 1 PAGE
N1,P.......START AT N1 & PAGE FORWARD
F,N1.......START AT N1 & PAGE BACK*
```

Interface 4. SEQ—Large Command Set

The structure of Interface 4 is nearly identical to that of Interface 3. The only difference is that with Interface 4 separate commands are provided for the various operations: with Interface 3 these were all done by using optional arguments to the DISPLAY command.

If the reader has reviewed and understood the structure and functions of Interface 3, it will be easy to see how the commands of Interface 4 work. Therefore, the reader should consult the description of Interface 4 for answers to questions about the basic operations available to the user of this Interface. The examples of Interface 4 commands will consequently be brief.

To retrieve a RANGE of items (for example, items 20-27) the user would enter: RANGE 20-27

As with Interface 3, various combinations can be used with the RANGE command. For example: RANGE 25,29-39,50-55,71

To display a particular item, say item 15, the user would enter: SHOW 15

To move back N items from the topmost item on the display, use the BACKUP command. If items 15-20 are on the display, then: BACKUP 5 would cause items 10-14 to be on the display. To move ahead N items from the bottom-most item, the JUMP command is used. Thus: JUMP 5 would cause items 26-30 to be displayed.

There are several "PAGE-type" commands to perform the paging operations described for Interface 3. For example, PPAGE will cause the preceding "page of items" (i.e., the preceding 12 items) to be shown. NPAGE causes the next page (12 items) to appear. FPAGE N and PAGEB N cause a page of items to be displayed starting with item N. For example FPAGE 30 causes items 30-42 to be displayed; PAGEB 30 causes items 18-30 to be displayed.

To automatically display the first 12 items in the list, the user can enter the command "FIRST." To automatically show the last 12 items in the database (i.e., the last "page"), the user would enter: END

To reveal the recipe of any item (for example item 15) the user would enter: RECIPE 15

The HELP file for Interface 4 is as follows:

```
RANGE N1-N2.....SHOW ITEMS N1-N2
SHOW N.........SHOW ITEM N
BACKUP N........GO BACK N ITEMS & SHOW
JUMP N.........JUMP AHEAD N ITEMS & SHOW
RECIPE N........SHOW RECIPE FOR ITEM N
PPAGE..........SHOW PRIOR PAGE
NPAGE.......... SHOW NEXT PAGE
FPAGE N.........PAGE FORWARD FROM N
PAGEB N.........PAGE BACK FROM N
FIRST..........START AT BEGIN; SHOW 1ST PAGE
END............SHOW LAST PAGE OF ITEMS*
```

Interface 5. MENU—Small Command Set, Single-Jump, Backup

Interface 5 uses a MENU-based, hierarchical structure. All main commands are options to one primary command, SELECT. In this interface the user can select just ONE item from the menu of choices displayed on the screen. The important concepts to understand are the following:

The user must continue to select items until a menu with ONLY ONE ITEM is produced. This is called a "Final Menu" and will consist of the name of exactly one recipe. The user can see the details of that recipe by entering: SELECT (i.e., the normal SELECT command but NOT followed by an item name). Since there is only one item in the menu, it is assumed that the user wants to see the recipe for that item.

When the user looks at a series of menus, the "last" or "previous" menu just seen may be either a higher-level menu or a lower-level menu. The top menu (i.e., the first menu displayed at the start of each problem) has the highest level, 1. The next level menu is level 2, etc. The BACKUP function always moves to the last-seen menu, whether that menu was higher or lower than the menu currently on the screen. Therefore, there is an important distinction between moving to the last-seen menu (BACKUP) and

moving to the "Next Higher Menu" (i.e., moving from a level 3 menu to a level 2 menu). (Of course, one moves to the "next lower level" by making selections from the choices shown on the current menu.)

The user is permitted to make a sequence of selections from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose the following menu is being shown:

1. Apple Pie
2. Cherry Pie
3. Peach Pie
4. Pecan Pie

The user can first select item 3, peach pie. This will cause the menu consisting of only peach pie to appear, i.e.: 1. Peach Pie

The user then enters SELECT to cause the recipe to appear. Now, suppose the user wants to see the recipe for Pecan Pie. This was item 4 on the previous menu. Instead of having to go back to the previous menu (for example, by using BACKUP), the user can directly request that the recipe for the "next menu item" be shown. If the next item is the name of a single recipe, then that recipe will automatically appear. If the next item represents several recipes, then a new menu will appear. For example, in the present case, since the next item is a single recipe (i.e., Pecan Pie), the user can move directly to that recipe by entering: SELECT +

Then the recipe for Pecan Pie would appear. But if the next item represented several recipes (e.g., Pecan Desserts) then a new menu would appear instead. Note that the user requests the "next menu item" but if it is at the bottom of the menu already, then an error should result.

Examples of command usage follow:

To select a choice (say, choice 3, Peach Pie) from the current menu, the user enters: SELECT Peach Pie

Note: ONLY one choice can be made. Otherwise an error should result. ALSO, note that choices are entered BY SPELLING OUT THE CHOICE, NOT by entering the Item Number. This is done to make the selection process similar to that used in the Keyword interfaces.

To choose next or previous item on a menu (as described above) the user enters: SELECT + or SELECT -

To go to the next higher level menu (for example if the user is viewing a level 3 menu, then to go to the preceding level 2 menu) the user enters: SELECT ^

And to go to the TOP menu (i.e., the level 1 menu), the user enters. SELECT T

To BACKUP to the last-seen or previous menu, the user enters:
SELECT P

The HELP file for Interface 5 is shown below:

```
SELECT (OPTION); WHERE OPTION IS:

ITEM1.......SELECT ITEM1 ON MENU
"BLANK".....SELECT ITEM & SHOW ITS RECIPE
+...........SELECT NEXT MENU ITEM
-...........SELECT PRIOR MENU ITEM
P...........GO TO PREVIOUS MENU
^...........GO TO NEXT HIGHER MENU
T...........GO TO TOP MENU*
```

Interface 6. MENU—Small Command Set, Single Jump, NO Backup

Interface 6 is identical to Interface 5 except that the BACKUP option (i.e., SELECT P) is not provided. Also the option to move to the "next higher menu" (SELECT ^) is not used. Therefore, no additional explanation is needed of the command options that are available.

The HELP file for Interface 6 is:

```
SELECT (OPTION); WHERE OPTION IS:

ITEM1.......SELECT ITEM1 ON MENU
"BLANK".....SELECT ITEM & SHOW ITS RECIPE
+...........SELECT NEXT MENU ITEM
-...........SELECT PRIOR MENU ITEM
T...........GO TO TOP MENU*
```

Interface 7. MENU - Small Command Set, Multi-Jump, Backup

Interface 7 is very similar to Interface 5. However, now users can make several item selections at once. For example, suppose the top menu shown below is being displayed:

1. Salads
2. Soups
3. Entrees
4. Desserts

Also, suppose that the user knows from experience that if Item 4, Desserts, is selected, the next menu presented will be:

1. Hot
2. Cold

Then instead of having to make two separate menu selections (e.g., 4, then 1) to see the selection of hot desserts, the user

car enter: SELECT Desserts,Hot and the level 3 menu consisting of hot dessert selections will be immediately presented without the intervening level 2 menu (Hot, Cold). This is what the "multi-jump" feature means. This feature is the only difference from Interface 5. The other command options function identically. The experimenter must be careful to make sure that NONE of the items specified in a multi-jump sequence is incorrect. It is also a little more difficult for the experimenter to keep track of previous menus etc.

With regard to BACKUP and next higher menu functions, the following rules should apply: when BACKUP is used, the last menu actually SEEN by the user is generated. For "next higher menu" the next higher menu level is shown even though this might not have been seen by the user (since the user could have bypassed it with a multi-level jump). For example, if the user begins with the TOP menu (level 1) and jumps directly to level 3 (as in the example above) then BACKUP would cause the top menu to be shown. But "next higher menu" (i.e., SELECT ^) would cause the level 2 menu (HOT,COLD) to appear.

The HELP file for Interface 7 is:

```
SELECT (OPTION); WHERE OPTION IS:

ITEM1,ITEM2.....SELECT ITEM1 FROM MENU; ITEM2 FROM NEXT
                MENU
"BLANK".........SELECT ITEM & SHOW ITS RECIPE
+...............SELECT NEXT MENU ITEM
-...............SELECT PRIOR MENU ITEM
P...............GO TO PREVIOUS MENU
^...............GO TO NEXT HIGHER MENU
T...............GO TO TOP PAGE*
```

Interface 8. MENU—Small Command Set, Multi-Jump, NO Backup

Interface 8 is identical to Interface 7 except that the BACKUP command option, SELECT P, is not available. Also, the command to move to the "next higher menu" (SELECT ^) is not available. All other features work exactly the same. The HELP file for Interface 8 is:

```
SELECT (OPTION); WHERE OPTIONS ARE:

ITEM1,ITEM2.....SELECT ITEM1 ON MENU; ITEM2 ON NEXT MENU
"BLANK".........SELECT ITEM & SHOW ITS RECIPE
+...............SELECT NEXT MENU ITEM
-...............SELECT PRIOR NEXT ITEM
T...............GO TO TOP MENU*
```

Interface 9. MENU—Large Command Set, Single-Jump, BACKUP

Interface 9 is similar in operation and structure to Interface 5. The only difference is that special commands are provided for each of the functions, rather than just options in the use of the SELECT command. The easiest way to understand Interface 9 is by comparing the commands with the options provided in Interface 5:

Interface 9 Command	Interface 5 Option
SELECT Item	SELECT Item
RECIPE	SELECT "blank"
NEXT	SELECT +
PRIOR	SELECT -
BACKUP	SELECT P
UP	SELECT ^
TOP	SELECT T

Thus, the meaning of all Interface 9 commands should be clear if the options of Interface 5 have been understood.

The HELP file for Interface 9 is:

```
SELECT ITEM1.....SELECT ITEM1 ON MENU
RECIPE.....SHOW RECIPE
NEXT.....SELECT NEXT MENU ITEM
PRIOR.....SELECT PRIOR MENU ITEM
BACKUP.....GO TO PREVIOUS MENU
UP.....GO TO NEXT HIGHER MENU
TOP.....GO TO TOP MENU*
```

Interface 10. MENU—Large Command Set, Single-Jump, NO Backup

Interface 10 is the same as Interface 9 except that the BACKUP command and the command to move to the "next higher menu" (UP) are not included.

The HELP file for Interface 10 is:

```
SELECT ITEM1.....SELECT ITEM1 ON MENU
RECIPE.....SHOW RECIPE
NEXT.....SELECT NEXT MENU ITEM
PRIOR.....SELECT PRIOR MENU ITEM
TOP.....GO TO TOP MENU*
```

Interface 11. MENU—Large Command Set, Multi-Jump, BACKUP

Interface 11 is identical in structure and function to Interface 7. The only difference is that specific commands have been used instead of options to the single command SELECT. A special "JUMP" command has also been added for the multi-jump condition (i.e., making multiple selections at one time). Now, the SELECT

command can only be used for making ONE selection at a time. The JUMP command MUST be used when making more than one selection at a time. (Remember, single and multi-jump selections both used SELECT in previous interfaces.)

The HELP file for Interface 11 is:

```
SELECT ITEM1.........SELECT "ITEM1" ON MENU
JUMP ITEM1,ITEM2.....SELECT ITEM1 ON MENU; ITEM2 ON NEXT
                     MENU
RECIPE...............SHOW RECIPE
BACKUP...............GO TO PREVIOUS MENU
UP...................GO TO NEXT HIGHER MENU
NEXT.................SELECT NEXT MENU ITEM
PRIOR................SELECT PRIOR MENU ITEM
TOP..................GO TO TOP MENU*
```

Interface 12. MENU--Large Command Set, Multi-Jump, NO Backup

Interface 12 is identical to Interface 11 except that the BACKUP command is no longer available. Also, the command to move to the "next higher level" (UP) has been omitted. The HELP file for Interface 12 is:

```
SELECT ITEM.........SELECT "ITEM" FROM MENU
JUMP ITEM1,ITEM2....SELECT ITEM1 ON MENU; ITEM2 ON NEXT
                    MENU
RECIPE..............SHOW RECIPE
NEXT................SELECT NEXT MENU ITEM
PRIOR...............SELECT PRIOR MENU ITEM
TOP.................GO TO TOP MENU*
```

APPENDIX C
Instructions

Interface 1. KW, Small

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning and food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to prepare a dinner which includes a fruit pie for dessert. Then you might want to get the recipe for apple pie.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipes to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the computer to get recipes. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful (but don't worry about forgetting the details of the problem—you can see the problem again whenever you wish).

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

The computer system gives you the following commands for getting information out of its storage files. These commands are:

```
SEARCH KW.....SEARCH FOR 'KW"
KW,D.....SEARCH FOR KW; SHOW MATCHES
```

KW,&,#.....SEARCH UNRESTRICTED; SHOW # OF MATCHES
NEW-LIST.....RESET SEARCH LIST
DISPLAY.....SHOW COMPLETE SEARCH LIST
FREE.....ELIMINATE SEARCH LIST FOR NEXT SEARCH
KW,RECIPE CHOICE...SHOWS THE MESSAGE "GOOD CHOICE" IF ONLY ONE MATCH IN SEARCH LIST
&,D.....SHOW ALL ITEMS IN LIST
PROBLEM.....SHOWS THE CURRENT PROBLEM
NEW PROBLEM.....SHOWS NEXT PROBLEM

Before giving a detailed explanation of these commands, it is important to understand the term "keyword". A "keyword" (abbreviated "KW") is just a word or group of letters that the computer tries to find in its storage files. Whenever it finds the keyword in a recipe name, that is called "finding a MATCH." For example, if you told the computer to look for a recipe containing the keyword "FISH" it would match the recipe for "BAKED FISH", "CODFISH BALLS", "FRIED FISH", etc. Keywords are the way you tell the computer what items you are looking for.

***** SEARCH *****

SEARCH will ONLY find the matching items and get them ready to display— it will NOT actually display them. In fact, it won't do ANYthing except locate the matching items and get them ready to display.

Try the following example:

Plan a dinner menu which includes fish as the main dish.

SEARCH FISH (followed by RETURN key)

You will see that the computer shows you that the command was carried out OK but does not show you any items. This is a safe way to find items if you do not know how many items might be matched. There is a special command-word to show how many items were found. Enter the following example:

&,# (followed by RETURN key)

Now the computer will tell you that your last SEARCH (looking for "FISH") found 12 items. Since this is not too many, you could safely ask the computer to display all the items it found. This will be explained more fully later.

The computer will now search and when the searching part is done, it will show how many items were found and are ready for display.

***** SEARCH LISTS *****

There are about 2000 recipes stored in the computer. However,

when looking for keywords, the computer does not always check ALL 2000 recipes. It only checks recipes in the current "search list". The search list is the list of items the computer thinks you are most interested in.

At first, the computer has no way of knowing which recipes you are most interested in, so it will begin by checking all 2000 recipes for the keyword you enter. So at first the search list is all 2000 recipes stored in the computer; but suppose you are interested in baked fish recipes.

You might first enter "SEARCH FISH". The computer will find all recipes containing the keyword "FISH". Suppose there are fifteen recipes containing this keyword. You will see the list of these items by using the command DISPLAY.

***** &,D *****

Displaying the Search List

The SEARCH command, when used alone, with just a keyword, simply searches for recipes containing the keyword and makes up a search list based on all the recipes it finds. It does nothing else. In particular, it does not display any items which it finds.

However, eventually you will want to see the items which the computer has found and put into the new search list. To display the items which the computer found and put into the current Search List, use the command-word DISPLAY. Enter the following example:

&,D (return)

This causes all the items in the current Search List to appear. (Remember, you made a Search List before by searching for FISH.) The display would look like this:

FISH IN A BASKET
MEAT FISH POULTRY
CODFISH BALLS
FRIED FISH
OVEN FRIED FISH
STEAMED FISH
FISH CHOWDER
MEAT POULTRY FISH
BROILED FISH
DEEP FRIED FISH
HERB BAKED FISH
SCALLOPED FISH
STUFFED FISH

You can always ask the computer to display all the items in the search list, but it is first a good idea to make sure there are not too many items in the list.

Notice that only some of the items in this list are actually main dish recipes. But now, if you search again using the keyword "FRIED", the computer will ONLY check the items you just found--items containing the word "FISH". So this time, the computer will check a much smaller "search list". The search list will consist only of the items you just found and these items all contain the word "FISH". When the computer checks this smaller search list, it will find only two items containing the new keyword "FRIED". These two items will then be combined into a new, even smaller search list as shown below:

FRIED FISH
OVEN FRIED FISH
DEEP FRIED FISH

Now the computer has found all the items containing the word "FISH" AND the word "FRIED". So you have successfully found all the fried fish recipes.

Notice that when you did the second search, looking for recipes containing the word "FRIED", the computer did not check recipes like steamed fish or codfish balls because these were NOT in the search list at the time the second search was made.

After the second search, since three items were found that matched your keyword, a new search list was made with only THREE items.

If NO recipes were found in the search list to match your keyword, THEN THE OLD SEARCH LIST (containing 13 FISH items) WOULD BE LEFT UNCHANGED. This is an important point to remember. The search list is only changed when items matching your keyword are actually found.

So you see that the computer gradually zeros in on the items you are looking for by making the search list smaller and smaller after each search. This is the basic idea of the "search list" concept.

***** NEW-LIST *****

Resetting the Search List

Perhaps you discover that you entered a poor keyword choice. Or perhaps you want to search for a completely different recipe. Then the small search list which the computer has formed may not contain the items you want to check. Instead you may want the computer to search through all 2000 recipes. To do this, you must "reset" the search list.

To do this, use the NEW-LIST command-word.

NEW-LIST (followed by return)

Now, the next time you request the computer to search for a keyword, all 2000 recipes will be checked and a new search list will be made up afterwards.

Normally, when the computer checks for items and finds matches, the OLD Search List is thrown out and a NEW Search List, based on the New matches, is made up. However, there are times when you might want to keep the current Search List and just ADD to it. The next command explains this.

***** FREE *****

Eliminate Search List for NEXT Search ONLY

Suppose you are looking for seafood dinners and have made up a search list containing the following:

FRIED FISH
OVEN FRIED FISH
DEEP FRIED FISH

These are all items you might want to use. But you also want to check for some additional items and add them to your Search List. Suppose you wanted to check to see if the computer had recipes for baked fish. If you searched for baked using the standard command:

SEARCH BAKED &,d

the computer would ONLY check your CURRENT Search List. Since the current list does NOT contain any baked items, no matches would be found. Or, you could get rid of your search list first with the NEW-LIST command-word. Then when you did SEARCH BAKED, the entire computer file would be checked and a new Search List would be made up of all baked items ... but then you would have lost your original Search List. This problem can be solved with the FREE command. This tells the computer that the next search should cover the ENTIRE storage file and that any matches should just be added to the current Search List without starting a new list. For example, try this:

NEW-LIST (return)
SEARCH FRIED (return)
FREE (return)
SEARCH BAKED (return)
DISPLAY (return)

BAKED FISH
DEEP FRIED FISH
FRIED FISH
HERB BAKED FISH
OVEN FRIED FISH

The only new command-word is FREE. The sequence you entered did the following: First, the current Search List was eliminated so the computer would check the entire storage file. Then the computer checked for all items containing the keyword FRIED. Then, you told the computer that the NEXT search should cover the entire storage file, not just the newly-made Search List (which contains all the items with BAKED in their name). Then, the next command-word said to find all items with BAKED and to ADD these items to the current search list. Finally, the last command-word told the computer to show the final Search List. As you can see, the final list contains BOTH FRIED items AND BAKED items.

To make a search list that contains only the item you are after, you should enter:

SEARCH HERB (return)

Now exactly one item is matched. You must still check to find if the recipe choice meets the needs of the problem. Since there is only one item in the search list, you can enter the command:

SEARCH HERB, RECIPE CHOICE (return)

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

Now you would write the name of the recipe on your answer worksheet and indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems, then enter:

DONE

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 2. KW, Large

This is an experiment to study how to make computers easier to use.

In this experiment, you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning and food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to prepare a dinner which includes a fruit pie for dessert. Then you might want to get the recipe for apple pie.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipes to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the computer to get recipes. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful (but don't worry about forgetting the details of the problem—you can see the problem again whenever you wish).

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special COMPLETE SEARCH LIST.

FREE.....ELIMINATE SEARCH LIST FOR NEXT SEARCH
RECIPE CHOICE.....SHOWS THE MESSAGE "GOOD CHOICE" IF ONLY
ONE MATCH IN SEARCH LIST
&.....WILDCARD CHARACTER*
PROBLEM.....SHOWS THE CURRENT PROBLEM
NEW PROBLEM.....SHOWS NEXT PROBLEM
DELETE KW.....EXCLUDE KW FROM NEXT SEARCH

NEW-LIST.....RESET SEARCH LIST
DISPLAY.....SHOW COMPLETE SEARCH LIST
FREE.....ELIMINATE SEARCH LIST FOR NEXT SEARCH
RECIPE CHOICE.....SHOWS THE MESSAGE "GOOD CHOICE" IF ONLY ONE MATCH IN SEARCH LIST
&.....WILDCARD CHARACTER*
PROBLEM.....SHOWS THE CURRENT PROBLEM
NEW PROBLEM.....SHOWS NEXT PROBLEM
DONE.....EXPERIMENT COMPLETED

Before giving a detailed explanation of these commands, it is important to understand the term "keyword". A "keyword" (abbreviated "KW") is just a word or group of letters that the computer tries to find in its storage files. Whenever it finds the keyword in a recipe name, that is called "finding a MATCH". For example, if you told the computer to look for a recipe containing the keyword "FISH" it would match the recipe for "BAKED FISH", "CODFISH BALLS", "FRIED FISH", etc. Keywords are the way you tell the computer what items you are looking for.

***** SEARCH *****

SEARCH will ONLY find the matching items and get them ready to display— it will NOT actually display them. In fact, it won't do ANYthing except locate the matching items and get them ready to display.

Try the following example:

Plan a dinner menu which includes fish as the main dish.

SEARCH FISH (followed by RETURN key)

You will see that the computer shows you that the command was carried out OK but does not show you any items. This is a safe way to find items if you do not know how many items might be matched. There is a special command-word to show how many items were found. Enter the following example:

SIZE (followed by RETURN key)

Now the computer will tell you that your last SEARCH (looking for "FISH") found 13 items. Since this is not too many, you could safely ask the computer to display all the items it found. This will be explained more fully later.

The computer will now search and when the searching part is done, it will show how many items were found and are ready for display.

***** SEARCH LISTS *****

There are about 2000 recipes stored in the computer. However, when looking for keywords, the computer does not always check ALL 2000 recipes. It only checks recipes in the current "search list". The search list is the list of items the computer thinks you are most interested in.

At first, the computer has no way of knowing which recipes you are most interested in so it will begin by checking all 2000 recipes for the keyword you enter. So at first the search list is all 2000 recipes stored in the computer. But suppose you are interested in baked fish recipes.

You might first enter "SEARCH FISH". The computer will find all recipes containing the keyword "FISH". Suppose there are fifteen recipes containing this keyword. You will see the list of these items by using the command DISPLAY.

***** DISPLAY *****

Displaying the Search List

The SEARCH command, when used alone, with just a keyword, simply searches for recipes containing the keyword and makes up a search list based on all the recipes it finds. It does nothing else. In particular, it does not display any items which it finds.

However, eventually you will want to see the items which the computer has found and put into the new search list. To display the items which the computer found and put into the current Search List, use the command-word DISPLAY. Enter the following example:

DISPLAY (return)

This causes all the items in the current Search List to appear. (Remember, you made a Search List before by searching for FISH.) The display would look like this:

FISH IN A BASKET
MEAT FISH POULTRY
CODFISH BALLS
FRIED FISH
OVEN FRIED FISH
STEAMED FISH
FISH CHOWDER
MEAT POULTRY FISH
BROILED FISH
DEEP FRIED FISH
HERB BAKED FISH
SCALLOPED FISH
STUFFED FISH

You can always ask the computer to display all the items in the search list, but it is first a good idea to make sure there are not too many items in the list.

Notice that only some of the items in this list are actually main dish recipes. But now, if you search again, using the keyword "FRIED" the computer will ONLY check the items you just found--items containing the word "FISH". So this time, the computer will check a much smaller "search list". The search list will consist only of the items you just found and these items all contain the word "FISH". When the computer checks this smaller search list, it will find only two items containing the new keyword "FRIED". These two items will then be combined into a new, even smaller search list as shown below:

FRIED FISH
OVEN FRIED FISH
DEEP FRIED FISH

Now the computer has found all the items containing the word "FISH" AND the word "FRIED". So you have successfully found all the fried fish recipes.

Notice, that when you did the second search, looking for recipes containing the word "FRIED", the computer did not check recipes like steamed fish or codfish balls because these were NOT in the search list at the time the second search was made.

After the second search, since three items were found that matched your keyword, a new search list was made with only THREE items.

If NO recipes were found in the search list to match your keyword, THEN THE OLD SEARCH LIST (containing 13 FISH items) WOULD BE LEFT UNCHANGED. This is an important point to remember. The search list is only changed when items matching your keyword are actually found.

So you see that the computer gradually zeros in on the items you are looking for by making the search list smaller and smaller after each search. This is the basic idea of the "search list" concept.

***** DELETE *****

Sometimes, you may wish to match all the items in the search list that do NOT contain a particular keyword. For example, suppose you want to see the recipes for FRIED FISH. But your search list contains other types of fried foods as well:

DEEP FRIED FISH
FRIED FISH

In this case, you might want to ELIMINATE the two fish items from the search list. You can do this with the DELETE command-word. If you enter:

DELETE DEEP (return)

the computer will check the search list and match all the items that do NOT contain the keyword FISH. So the search list would end up as:

FRIED FISH
OVEN FRIED FISH

With the DELETE command-word, whenever a match is made, the matching item is deleted or removed from the current search list.

***** NEW-LIST *****

Resetting the Search List

Perhaps you discover that you entered a poor keyword choice. Or perhaps you want to search for a completely different recipe. Then the small search list which the computer has formed may not contain the items you want to check. Instead you may want the computer to search through all 2000 recipes. To do this, you must "reset" the search list.

To do this, use the NEW-LIST command-word.

NEW-LIST (followed by return)

Now, the next time you request the computer to search for a keyword, all 2000 recipes will be checked and a new search list will be made up afterwards.

Normally, when the computer checks for items and finds matches, the OLD Search List is thrown out and a NEW Search List, based on the New matches, is made up. However, there are times when you might want to keep the current Search List and just ADD to it. The next command explains this.

***** FREE *****

Eliminate Search List for NEXT Search ONLY

Suppose you are looking for seafood dinners and have made up a search list containing the following:

FRIED FISH
OVEN FRIED FISH
DEEP FRIED FISH

These are all items you might want to use. But you also want to check for some additional items and add them to your Search List. Suppose you wanted to check to see if the computer had recipes for baked fish. If you searched for baked using the standard command:

SEARCH BAKED

the computer would ONLY check your CURRENT Search List. Since the current list does NOT contain any baked items, no matches would be found. Or, you could get rid of your search list first with the NEW-LIST command-word. Then when you did SEARCH BAKED, the entire computer file would be checked and a new Search List would be made up of all baked items..but then you would have lost your original Search List. This problem can be solved with the FREE command. This tells the computer that the next search should cover the ENTIRE storage file and that any matches should just be added to the current Search List without starting a new list. For example, try this:

NEW-LIST (return)
SEARCH FRIED (return)
FREE (return)
SEARCH BAKED (return)
DISPLAY (return)
BAKED FISH
DEEP FRIED FISH
FRIED FISH
HERB BAKED FISH
OVEN FRIED FISH

The only new command-word is FREE. The sequence you entered did the following: First, the current Search List was eliminated so the computer would check the entire storage file. Then the computer checked for all items containing the keyword FRIED. Then, you told the computer that the NEXT search should cover the entire storage file, not just the newly-made Search List (which contains all the items with BAKED in their name). Then, the next command-word said to find all items with BAKED and to ADD these items to the current search list. Finally, the last command-word told the computer to show the final Search List. As you can see, the final list contains BOTH FRIED items AND BAKED items.

To make a search list that contains only the item you are after, you should enter:

SEARCH HERB (return)

Now exactly one item is matched. You must still check to find if the recipe choice meets the needs of the problem. Since there is only one item in the search list you can enter the command:

RECIPE CHOICE (return)

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

Now you would write the name of the recipe on your answer

worksheet and indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems, then enter:

DONE

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 3. Sequential, Small

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning and food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to prepare a dinner which includes a fruit pie for dessert. Then you might want to get the recipe for apple pie.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipes to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the computer to get recipes. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful (but don't worry about forgetting the details of the problem—you can see the problem again whenever you wish).

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

The structure of the experiment is a simple numerical sequence. The database simply seems to be an ordered list of recipe names. Thus, your database view is a linear sequence. The commands provided in this database allow you to retrieve one or more items by specifying their sequence numbers. A range of items (e.g., 17-34) can also be retrieved.

The HELP file is shown below:

```
DISPLAY (OPTION); WHERE OPTION IS:

N1-N2.............SHOW ITEMS IN RANGE N1-N2
N1................SHOW ITEM N1
-N1...............MOVE BACK N1 ITEMS
+N1...............MOVE AHEAD N1 ITEMS
N1,N2-N3..........SHOW ITEMS N1 & N2-N3
N,RECIPE CHOICE...PRESENTS THE MESSAGE "GOOD CHOICE"
-P................DISPLAY BACK 1 PAGE
+F................DISPLAY FORWARD 1 PAGE
N1,P..............START AT N1 & PAGE FORWARD
P,N1..............START AT N1 & PAGE BACK*
```

The only other basic concept is that of a PAGE of items. This means 15 or so items that are grouped together. For example, if items 15-30 are currently being displayed, you can request the next Page of items, (e.g., 31-45) or the previous Page (1-15).

All the commands are basically options done with the single command: DISPLAY. The various operations that can be done with the DISPLAY command are as follows:

To display a single item (for example item number 27) enter:

DISPLAY 27 (press return)

To display a range of items (say, items 27-35) enter:

DISPLAY 27-35 (press return)

You could also specify both a single item (or items) as well as a range (or ranges). For example, to retrieve and display item 3,6, 15-20, 27, and 30-35, the user could enter:

DISPLAY 3,6,15-20,27,30-35 (PRESS RETURN)

Make sure the item numbers are entered properly and do not specify too large a range. For example, the following commands should result in errors:

DISPLAY 29880
DISPLAY 35-27
DISPLAY 1-500

If the top-most item currently being displayed is number X (e.g., if items 25-32 are being displayed, then X=25), then you can move back N items (for example 23 items) by the following command:

DISPLAY -23 (press return)

Since 7 items were on the screen, now, after moving back 23 items, the screen should show 7 new items, 2-9 (i.e., the new

display begins with 25-23=2 and shows 7 items). Similarly, you could jump ahead from the bottom-most number on the display screen. For example, if items 25-32 were being shown, the user could jump ahead 5 items by entering:

DISPLAY +5 (press return)

This would cause items 30-37 to be on the display screen. Make sure that you do not enter a command that would go past either end of the database sequence. For example, if items 3-10 were being displayed, the following commands should be errors:

DISPLAY -15
DISPLAY +9997

To get the message "Good Choice" you enter, for example:

DISPLAY 15,RECIPE CHOICE (press return)

This command will work when there is only one recipe in the search list.

To move ahead or back one page from the currently displayed items, you could enter "DISPLAY +8 or DISPLAY -10" for example. However, to automatically move ahead or back 12 items, use the "Page" option as follows:

DISPLAY -P (press return) or DISPLAY +P (press return)

Thus, if items 20-30 were on the display, then "DISPLAY -P" would cause items 8-20 to be displayed; "DISPLAY +P" would cause items 30-32 to be displayed. Note that a "full page" of items is always displayed; even though only 10 items were being shown, when the "Display Next/Previous Page" option is used, the next or previous FULL page of items is shown. If this goes beyond the limit of the database, then the message "No More Pages" or an equivalent should be given. (For example, if items 1-12 are being displayed, DISPLAY -P should result in the "No More Pages" response. If items 3-10 are being displayed, the DISPLAY -P should result in items 1-12 being displayed.)

You may also specify a starting point at which the paging operation should begin. For example, to show the 12-item page which BEGINS with item 27 (i.e., items 27-39) enter:

DISPLAY 27,P (press return)

But to display the page ENDING with item 27 (i.e., items 15-27) enter:

DISPLAY P,27 (press return)

Now exactly one item is matched. You must still check to find if the recipe choice meets the needs of the problem. Since there is only one item in the search list, you can enter the command:

N,RECIPE CHOICE

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

Now you would write the name of the recipe on your answer worksheet and indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems, then enter:

DONE

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 4. Sequential, Large

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning and food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to prepare a dinner which includes a fruit pie for dessert. Then you might want to get the recipe for apple pie.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipes to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the computer to get recipes. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful (but don't worry about forgetting the details of the problem--you can see the problem again whenever you wish).

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

The following commands must be used to retrieve the items:

DISPLAY (OPTION); WHERE OPTION IS:

RANGE N1-N2.....SHOW ITEMS N1-N2
SHOW N.....SHOW ITEM N

```
BACKUP N.....GO BACK N ITEMS & SHOW
JUMP N.....JUMP AHEAD N ITEMS & SHOW
N,RECIPE CHOICE....PRESENTS THE MESSAGE "GOOD CHOICE"
PPAGE.....SHOW PRIOR PAGE
NPAGE.....SHOWS NEXT PAGE
FPAGE N.....PAGE FORWARD FROM N
PAGEB N.....PAGE BACK FROM N
FIRST.....START AT BEGINNING; SHOW 1ST PAGE
END.....SHOW LAST PAGE OF ITEMS*
```

The structure of the experiment is a simple numerical sequence. The database simply seems to be an ordered list of recipe names. Thus, your database view is a linear sequence. The commands provided in this database allow you to retrieve one or more items by specifying their sequence numbers. A range of items (e.g., 17-34) can also be retrieved.

To retrieve a range of items (for example, items 20-27) enter:

RANGE 20-27

Various combinations can be used with the RANGE command. For example:

RANGE 25,29-39,50-55,71

To display a particular item, say item 15, you would enter:

SHOW 15

To move N items from the topmost item on the display, use the BACKUP command. If items 15-20 are on the display, then:

BACKUP 5

would cause items 10-14 to be on the display. To move ahead N items from the bottom-most item, the JUMP command is used. Thus,

JUMP 5

would cause items 26-30 to be displayed.

There are several "PAGE-type" commands to perform the paging operations. For example, PPAGE will cause the preceding page of items (i.e., the preceding 12 items) to be shown. NPAGE causes the next page (12 items) to appear. FPAGE N and PAGEB N cause a page of items to be displayed starting with item N. For example, FPAGE 30 causes items 30-42 to be displayed; PAGEB 30 causes items 10-30 to be displayed.

To automatically display the first 12 items in the list, you can enter the command "FIRST". To automatically show the last 12 items in the database (i.e., the last page), enter END.

To get the message "Good Choice" enter for example:

SHOW 15,RECIPE CHOICE

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

Now you would write the name of the recipe-choice on your answer worksheet and indicate you are ready for the next problem or next part of the problem. When you retrieve all the items required then enter:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems then enter:

DONE

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 5. Single Jump, Small, Backup

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning and food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to plan a dinner which includes fish, vegetable, and some form of potato.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipe choices to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the recipe database to solve the problem. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful (but don't worry about forgetting the details of the problem—you can see the problem again whenever you wish).

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

For example, the problem is to plan a dinner menu that includes fish.

Following is the list of commands you will be using to solve all the problems.

SELECT (OPTION); WHERE OPTION IS:

T.....GO TO TOP MENU
ITEM NAME.....SELECT ITEM ON MENU
+.....SELECT NEXT MENU ITEM
-.....SELECT PRIOR MENU ITEM
P.....GO TO PREVIOUS MENU
H.....GO TO NEXT HIGHER MENU
ITEM NAME,RECIPE CHOICE...GIVES YOU THE MESSAGE "GOOD CHOICE"

Just to get started, when you see ENTER COMMAND: at the bottom of your screen, you will use the command:

SELECT T (return)

The following menu will then be displayed on your monitor:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

You can then choose the one that you think will be most helpful in solving the problem. In this case, you would enter:

SELECT MAIN DISHES (return)

The computer will then display the following sub-menu on its screen:

MAIN DISHES
BREAKFAST LUNCH DINNER

Your choice from this menu would be:

SELECT DINNER (return)

because you are searching for a dinner menu. The computer will now give the next display:

MAIN DISHES (DINNER)
VEGETARIAN NON-VEGETARIAN

Since you are looking for a menu that contains fish, your next response would be:

SELECT NON-VEGETARIAN (return)

The computer still has another sub-menu in the recipe data-base, and it will print out:

MAIN DISHES (DINNER, NON-VEGETARIAN)
MEAT POULTRY SEAFOOD

Your next response would be:

SELECT SEAFOOD (return)

You will get another sub-menu:

SEAFOOD FISH TUNA

You will enter:

SELECT FISH (return)

Immediately after that the computer will give you a list of items containing Fish:

BROILED FISH
CODFISH BALL
DEEP FRIED FISH
FLOUNDER PROVENCALE
FRIED FISH
HADDOCK SHRIMP BAKE
HALIBUT ROYALE
HERB BAKED FISH
OVEN FRIED FISH
SALMON LOAF
SALMON STEAK
SCALLOPED FISH
STEAMED FISH
STUFFED FLOUNDER
STUFFED WHITEFISH
FISH IN A BASKET
GRILLED HALIBUT
HICKORY FISH BAKED
FILLETS ELEGANTE

The final menu is made by choosing a name on the database list—for example SELECT DEEP FRIED FISH. This is called a "FINAL menu" and will consist of the name of exactly one recipe.

When you look at a series of menus, the "last" or "previous" menu just seen may be either a higher level menu or a lower level menu. The top menu (i.e., the first menu displayed at the start of each problem) has the highest level, 1. The next level menu is level 2, etc. The BACKUP function always moves to the last seen menu, whether that menu was higher or lower than the menu currently on the screen. Therefore, there is an important distinction between moving to the last seen menu (BACKUP) and moving to the "next higher menu" (i.e., moving from a level 3 menu to a level 2 menu). Of course, one moves to the "next lower level" by making selections from the choices shown on the current menu.

You are permitted to make a sequence of selections from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose you had selected item 3, Deep Fried Fish, from the recipe database list of fish, above. This would cause the menu consisting only of deep fried fish to appear, i.e.:

DEEP FRIED FISH

Now, suppose that you want to see the recipe for Flounder Provencale. This was item 4 on the previous menu. Instead of having to go back to the previous menu (for example, by using BACKUP), you can directly request that the recipe for the "next menu item" be shown. If the next item is the name of a single recipe, then that recipe will automatically appear. If the next item represents several recipes, then a new menu will appear. For example, in the present case, since the next item is a single recipe (i.e., flounder provencale), you can move directly to that recipe by entering:

SELECT + (return)

Then the recipe for flounder provencale would appear. But if the next item represented several recipes (e.g., Fish), then a new menu would appear instead. Note that if you request the "next menu item" but are already at the bottom of the menu, then an error will result.

Examples of command usage follow:

To select a choice (say, choice 3, deep fried fish) from the current menu, enter:

SELECT DEEP FRIED FISH,RECIPE CHOICE (return)

Note: Only one choice can be made. Otherwise an error should result. Also note that choices are entered BY SPELLING OUT THE CHOICE, NOT by entering the item number.

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

To choose the next or previous item on a menu (as described above) enter:

SELECT + (return)
or
SELECT - (return)

To go to the next higher level menu (for example, if you are using a level 3 menu, then to go to the preceding level 2 menu), enter:

SELECT H (return)

And to go to the top menu (i.e., the level 1 menu), enter:

SELECT T (return)

To BACKUP to the last seen or previous menu, the user enters:

SELECT P (return)

Now you would write the name of the recipe on your answer worksheet and indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems then enter:

DONE (return)

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 6. Single Jump, Small, No Backup

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning using food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to plan a dinner which includes fish, vegetable, and some form of potato.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipes to solve the problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the recipe database to solve the problem. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful (but don't worry about forgetting the details of the problem--you can see the problem again whenever you wish by entering the command "PROBLEM" (return)).

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

For example, the problem is to plan a dinner menu that includes fish.

Following is the list of commands you will be using to solve all the problems.

SELECT (OPTION); WHERE OPTION IS:

T.....GO TO TOP MENU
ITEM NAME.....SELECT ITEM ON MENU
+.....SELECT NEXT MENU ITEM
-.....SELECT PRIOR MENU ITEM
ITEM NAME,RECIPE CHOICE....GIVES YOU THE MESSAGE "GOOD CHOICE"

Just to get started, when you see ENTER COMMAND: at the bottom of your screen, you will use the command:

SELECT T (return)

The following menu will then be displayed on your monitor:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

You can then choose the one that you think will be most helpful in solving the problem. In this case, you would enter:

SELECT MAIN DISHES (return)

The computer will then display the following sub-menu on its screen:

MAIN DISHES
BREAKFAST LUNCH DINNER

Your choice from this menu would be:

SELECT DINNER (return)

because you are searching for a dinner menu. The computer will now give the next display:

MAIN DISHES (DINNER)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Since you are looking for a menu that contains fish, your next response would be:

SELECT SEAFOOD (return)

The computer still has sub-menus in the recipe database, and it will print out:

MAIN DISHES (DINNER,SEAFOOD)
SEAFOOD FISH TUNA

Since the problem had asked for a recipe containing fish, your next choice could be:

SELECT FISH (return)

and the computer will give you a listing of the recipes it has in the database listed under fish:

BROILED FISH
CODFISH BALL
DEEP FRIED FISH
FLOUNDER PROVENCALE
FRIED FISH
HADDOCK SHRIMP BAKE
HALIBUT ROYALE
HERB BAKED FISH
OVEN FRIED FISH
SALMON LOAF
SALMON STEAK
SCALLOPED FISH
STEAMED FISH
STUFFED FLOUNDER
STUFFED WHITEFISH
FISH IN A BASKET
GRILLED HALIBUT
HICKORY FISH BAKED
FILLETS ELEGANTE

The final menu is made by choosing a name from the database list—for example, SELECT DEEP FRIED FISH. This is called a "FINAL menu" and will consist of the name of exactly one recipe.

You are permitted to make a sequence of selections from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose you had selected item 3, Deep Fried Fish, from the recipe database list of fish, above. This would cause the menu consisting only of deep fried fish to appear, i.e.:

DEEP FRIED FISH

Now, suppose that you want to see the recipe for Flounder Provencale. This was item 4 on the previous menu. You can directly request that the recipe for the "next menu item" be shown. If the next item is the name of a single recipe, then that recipe will automatically appear. If the next item represents several recipes, then a new menu will appear. For example, in the present case, since the next item is a single recipe (i.e., flounder provencale), you can move directly to that recipe by entering:

SELECT + (return)

Then the recipe for flounder provencale would appear. But if the next item represented several recipes (e.g., Fish) then a new menu would appear instead. Note that if you request the "next menu item" but are already at the bottom of the menu, then an error will result.

Suppose you make a mistake and choose the wrong item from a menu. For example, maybe you chose "Desserts" when you really wanted "Main Dishes". You can always "start over" completely by going back directly to the top menu. To do this at any time, just enter:

SELECT T (return)

Just remember that the "SELECT T" option always lets you start over from the first (top) menu.

Examples of command usage follow:

To select a choice (say, choice 3, deep fried fish) from the current menu, enter:

SELECT DEEP FRIED FISH,RECIPE CHOICE (return)

Note: Only one choice can be made. Otherwise an error should result. Also note that choices are entered BY SPELLING OUT THE CHOICE, NOT by entering the item number.

To choose the next or previous item on a menu (as described above) enter:

SELECT + (return)
or
SELECT - (return)

And to go to the top menu (i.e., the level 1 menu), enter:

SELECT T (return)

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

Now you would write the name of the recipe on your answer worksheet and indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems then enter:

DONE (return)

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier

as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 7. Multiple, Small, Backup

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning using food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to plan a dinner which includes fish, a vegetable, and some form of potato.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipe choices to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the recipe database to solve the problem. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful. (But don't worry about forgetting the details of the problem--you can see the problem again whenever you wish by entering the command "PROBLEM" [return].)

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

For example, the problem is to plan a dinner menu that includes fish.

Following is the list of commands you will be using to solve all the problems.

SELECT (OPTION); WHERE OPTION IS:

ITEM1.....SELECT ITEM1 ON MENU
ITEM1, ITEM2...SELECT ITEM1 ON MENU1, THEN ITEM ON MENU2, ETC.
RECIPE CHOICE..PRESENTS THE MESSAGE "GOOD CHOICE"
P.....GO TO PREVIOUS MENU
H.....GO TO NEXT HIGHER MENU
T.....GO TO TOP MENU
+.....STANDS FOR NEXT CHOICE
-.....STANDS FOR PREVIOUS CHOICE

Just to get started, when you see ENTER COMMAND: at the bottom of your screen, you will use the command:

SELECT T (return)

The following menu will then be displayed on your monitor:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

You can then choose the one that you think will be most helpful in solving the problem. In this case, you enter:

SELECT MAIN DISHES (return)

The computer will then display the following sub-menu on its screen:

MAIN DISHES
BREAKFAST LUNCH DINNER

Your choice from this menu would be:

SELECT DINNER (return)

because you are searching for a dinner menu. The computer will now give the next display:

MAIN DISHES (DINNER)
VEGETARIAN NON-VEGETARIAN

You choose non-vegetarian because you want a fish recipe. The next display would be:

MAIN DISHES (DINNER, NON-VEGETARIAN)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Since you are looking for a menu that contains fish, your next response would be:

SELECT SEAFOOD (return)

The computer still has sub-menus in the recipe database, and it will print out:

MAIN DISHES (DINNER, NON-VEGETARIAN, SEAFOOD)
SEAFOOD FISH TUNA

Since the problem had asked for a recipe containing fish, your next choice could be:

SELECT FISH (return)

and the computer will give you a listing of the recipes it has in the database listed under fish:

BROILED FISH
CODFISH BALL
DEEP FRIED FISH
FLOUNDER PROVENCALE
FRIED FISH
HADDOCK SHRIMP BAKE
HALIBUT ROYALE
HERB BAKED FISH
OVEN FRIED FISH
SALMON LOAF
SALMON STEAK
SCALLOPED FISH
STEAMED FISH
STUFFED FLOUNDER
STUFFED WHITEFISH
FISH IN A BASKET
GRILLED HALIBUT
HICKORY FISH BAKED
FILLETS ELEGANTE

The final menu is made by choosing a name from the database list—for example, SELECT DEEP FRIED FISH. This is called a "FINAL MENU" and will consist of exactly one recipe.

You are permitted to make a sequence of selections from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose you had selected item 3, Deep Fried Fish, from the recipe database list of fish, above. This would cause the menu consisting only of deep fried fish to appear, i.e.:

DEEP FRIED FISH (return)

Now, suppose that you want to see the recipe for Flounder Provencale. This was item 4 on the previous menu. You can directly request that the recipe for the "next menu item" be shown. If the next item is the name of a single recipe, then that recipe will automatically appear. If the next item represents several recipes, then a new menu will appear. For example, in the present case, since the next item is a single recipe (Flounder Provencale), you can move directly to that recipe by entering:

SELECT + (return)

Then the recipe for Flounder Provencale would appear. But if the next item represented several recipes (e.g., Fish) then a new menu would appear instead. Note that if you request the "next menu item" but are already at the bottom of the menu, then an error will result. To choose the previous item on a list, enter:

SELECT - (return)

ITEM 1, ITEM 2...

This command can be used to select items from more than one menu. For example, you want a dinner menu to solve a problem. You will first use the command SELECT T to get to the top menu. Then, out of that menu you will select Main Dishes and after that you will select the dinner option. The command ITEM1, ITEM2... allows you to make the selection from different levels simultaneously. For example, after you get the top menu, you can use this command by typing:

SELECT MAIN DISHES, DINNER (return)

This will give you the dinner menu. This will only work if you remember what the next menu is going to be. Otherwise the computer will you an error message.

How to Correct Some Mistakes

If you make a mistake and ask for something that is not on the menu, the computer will tell you that it can't figure out what you want and will ask you to re-enter your choice. For example, if you entered SELECT CLD, the computer would not be able to find what you wanted so you would have to re-enter.

Your main objective in finding an item you want is to keep selecting until you get down to a menu with just <u>one</u> item. Suppose you make a mistake and choose the wrong item from a menu. For example, maybe you chose "MAIN DISHES" when you really wanted "DESSERTS". You can "go back" in three ways. These are each slightly different so it is important to carefully understand the three ways of correcting mistakes.

First, you can always "start over" completely by going back directly to the top menu. To do this at any time, just enter:

SELECT T (return)

Second, you can always go back to the menu you just saw (i.e., the last menu you saw before the one that is now on the screen), by using the P (for Previous menu) option:

SELECT P (return)

If you keep using the SELECT P option several times in a row, you will see that you just keep going back and forth between the same two menus (because one was always the "last" menu you saw before the one that is on the screen). To see this, try entering SELECT P several times.

You can see that the first time you enter SELECT P you went back to the top menu. The next time you went back to the "Breakfast, Lunch, Dinner" menu. So the P option lets you go both "forward" (i.e., to the next level down--the "Breakfast, Lunch, Dinner" menu is one level down below the top menu) and "backward" (i.e., go to the top menu from the "Breakfast, Lunch, Dinner" menu).

The last way to go back to a menu and correct a choice is to use the H option (for Higher level). This always takes you back to the menu one step above the menu on the screen. For example, if the "Dessert-type" menu list was on the screen (e.g., Drop Cookies), then entering:

SELECT H (return)

will get you back to the top menu (Desserts, Cookies; Bar, Drop, Filled, Shaped). Then if you entered SELECT H again, this would get you to the top menu (DESSERTS; CAKES, CANDIES, COOKIES, CUSTARDS, FROZEN, FRUIT, PIES, PUDDINGS) since this is the menu above the (DESSERTS, COOKIES; BAR, DROP, FILLED, SHAPED).

If you get to the top menu, SELECT H will not have any further effect since the top menu is as far as you can go.

Notice that SELECT H keeps moving you up toward the top menu. But SELECT P would just keep moving you between two menu levels. So the SELECT P option is good for moving back one menu (in either direction) while the SELECT H option is best for moving one or more levels back toward the top menu.

Once you get to the menu you want, you can just select a new choice from it.

How to Get the Message "Good Choice"

Normally, you should keep making menu choices until you narrow the menus to one recipe item--the recipe you want to choose for your menu plan. Use the "Recipe Choice" option by typing:

SELECT FRIED FISH, RECIPE CHOICE (return)

After you receive the message "GOOD CHOICE", you can continue in several ways. You could enter SELECT T, for example, to start over by going back to the top menu. You could also type SELECT P or SELECT H to return to the previous menu you were looking at.

Note: Only choice can be made. Otherwise an error should result.

Also note that choices are entered BY SPELLING OUT THE CHOICE, NOT by entering the item number.

Combining Options

Options such as Recipe Choice and P can be combined to make the SELECT command work more effectively. For example, suppose you just got the message "GOOD CHOICE", but you decide against that choice and want to choose a seafood option. One way to do this is to type SELECT P to backup to the last menu you saw, and then SELECT H to go up one more menu, to the next higher menu.

An easier way to do this is to to combine the options and just type:

SELECT P,H,+,RECIPE CHOICE (return)

You should be able to see how these options have been combined. The only new thing is the "+" sign. This means "Next Item in Menu". You could have typed:

SELECT P,+,RECIPE CHOICE (return)

but since you know that you just wanted to check the next item in the menu, the "+" symbol was used instead of the full name. This is helpful especially if you cannot remember what the exact name of the next menu item was. You also can use the "-" symbol to mean the "previous" item on the menu. For example, if you had chosen COD FISH BALLS SELECT P,H,-,RECIPE CHOICE [return] would give you the message "GOOD CHOICE"

Here is another example of combining options. Suppose you had a menu that contained several items. Normally, you would first type SELECT COD FISH BALLS to choose this item and form a new menu with just one item--COD FISH BALLS. Then you would use the SELECT RECIPE CHOICE option to get the message "GOOD CHOICE". However, you could combine these steps by typing the combination shown here:

SELECT COD FISH BALLS,RECIPE CHOICE (return)

You can see that this first selected COD FISH BALLS, RECIPE CHOICE and then displayed the message "GOOD CHOICE".

Remember, you can combine options in any other way that you think will help you get to the information you want more easily.

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

Now you would write the name of the recipe on your answer worksheet and indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems, then enter:

DONE (return)

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by the "return" key)

Instructions

Interface 8. Multiple, Small, No Backup

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning using food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to plan a dinner which includes fish, a vegetable, and some form of potato.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipe choices to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the recipe database to solve the problem. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful. (But don't worry about forgetting the details of the problem--you can see the problem again whenever you wish by entering the command "PROBLEM" [return].)

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

For example, the problem is to plan a dinner menu that includes fish.

Following is the list of commands you will be using to solve all the problems.

SELECT (OPTION); WHERE OPTION IS:

T.....GO TO TOP MENU
ITEM NAME.....SELECT ITEM ON MENU
+.....SELECT NEXT MENU ITEM
-.....SELECT PRIOR MENU ITEM
ITEM1, ITEM2.....SELECT ITEM1 ON MENU1, THEN ITEM2 ON MENU2, ETC.
ITEM NAME, RECIPE CHOICE..GIVES YOU THE MESSAGE "GOOD CHOICE"

Just to get started, when you see ENTER COMMAND: at the bottom of your screen, you will use the command:

SELECT T (return)

The following menu will then be displayed on your monitor:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

You can then choose the one that you think will be most helpful in solving the problem. In this case, you would enter:

SELECT MAIN DISHES (return)

The computer will then display the following sub-menu on its screen:

MAIN DISHES
BREAKFAST LUNCH DINNER

Your choice from this menu would be:

SELECT DINNER (return)

MAIN DISHES (DINNER)
VEGETARIAN NON-VEGETARIAN

Now, your choice from this menu would be:

SELECT NON-VEGETARIAN (return)

The computer would display:

MAIN DISHES (DINNER, NON-VEGETARIAN)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Since you are looking for a menu that contains fish, your next response would be:

SELECT SEAFOOD (return)

The computer still has sub-menus in the recipe database, and it will print out:

MAIN DISHES (DINNER, SEAFOOD)
SEAFOOD FISH TUNA

Since the problem had asked for a recipe containing fish, your next choice could be:

SELECT FISH (return)

and the computer will give you a listing of the recipes it has in the database listed under fish:

BROILED FISH
CODFISH BALL
DEEP FRIED FISH
FLOUNDER PROVENCALE
FRIED FISH
HADDOCK SHRIMP BAKE
HALIBUT ROYALE
HERB BAKED FISH
OVEN FRIED FISH
SALMON LOAF
SALMON STEAK
SCALLOPED FISH
STEAMED FISH
STUFFED FLOUNDER
STUFFED WHITEFISH
FISH IN A BASKET
GRILLED HALIBUT
HICKORY FISH BAKED
FILLETS ELEGANTE

The final menu is made by choosing a recipe from the above list--for example, SELECT DEEP FRIED FISH. This is called a "FINAL MENU" and will consist of the name of exactly one recipe.

You are permitted to make a sequence of selections from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose you had selected item 3, Deep Fried Fish, from the recipe database list of fish, above. This would cause the menu consisting only of deep fried fish to appear, i.e.:

DEEP FRIED FISH (return)

Now, suppose that you want to see the recipe for Flounder Provencale. This was item 4 on the previous menu. You can directly request that the recipe for the "Next Menu Item" be shown. If the next item is the name of a single recipe, then that recipe will automatically appear. If the next item represents several recipes, then a new menu will appear. For example, in the present case, since the next item is a single recipe (i.e., Flounder Provencale), you can move directly to that recipe by entering:

SELECT + (return)

The recipe for Flounder Provencale would appear. But if the next item represented several recipes (e.g., Fish) then a new menu would appear instead. Note that if you request the "Next Menu Item" when already at the bottom of the menu, then an error will result.

Similarly, if you want to choose Codfish Ball you can use the command

SELECT - (return)

ITEM1, ITEM2...

This command can be used to select items from more than one menu. For example, if you want dinner menu to solve a problem you will first use the command SELECT T to get the top menu. Then out of that menu you will select Main Dishes. When you get the menu

MAIN DISHES (BREAKFAST, LUNCH, DINNER)

only then you can select the dinner option. The command ITEM1, ITEM2... allows you to make the selection from different levels simultaneously. For example, after you got the top menu DESSERTS, MAIN DISHES, SIDE DISHES... you can use the command by typing:

SELECT MAIN DISHES, DINNER (return)

This will give you the dinner menu. This will only work if you remember what the next menu is going to be; otherwise the computer will give you an error message.

When you finally choose the item you want for your menu, you can use the option with an item choice, for example:

SELECT DEEP FRIED FISH, RECIPE CHOICE (return)

Note: Only one choice can be made. Otherwise an error should result.

Also note that choices are entered BY SPELLING OUT THE CHOICE, NOT by entering the item number.

If the choice is appropriate, the computer will give you the message: "GOOD CHOICE!"

Now you would write the name of the recipe on your answer worksheet. If you have completed all the segments of the problem, indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems then enter:

DONE (return)

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 9. Single, Large, Backup

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning using food recipes. So the information you will be trying to get will be names of different food dishes. For example, you might be asked to plan a dinner which includes fish, a vegetable, and some form of potato.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipe choices to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the recipe database to solve the problem. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful. (But don't worry about forgetting the details of the problem; you can see the problem again whenever you wish by entering the command "PROBLEM" [return].)

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO to computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

For example, the problem is to plan a dinner menu that includes fish.

Following is the list of commands you will be using to solve all the problems.

SELECT (OPTION); WHERE OPTION IS:

GET ITEM1.....GET ITEM1 ON MENU
RECIPE CHOICE..PRESENTS THE MESSAGE "GOOD CHOICE"
NEXT.....SELECT NEXT MENU ITEM AND DISPLAY
PRIOR.....SELECT PRIOR MENU ITEM AND DISPLAY
BACKUP.....GO TO PREVIOUS MENU
UP.....GO TO NEXT HIGHER MENU
TOP.....GO TO TOP MENU

Just to get started, when you see ENTER COMMAND: at the bottom of your screen, you will use command:

TOP (return)

Then following menu will then be displayed on your monitor:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

You can then choose the one that you think will be most helpful in solving the problem. In this case, you would enter:

GET MAIN DISHES (return)

The computer will then display the following sub-menu on its screen:

MAIN DISHES
BREAKFAST LUNCH DINNER

Your choice from this menu would be:

GET DINNER (return)

because you are searching for a dinner menu. The computer will now give you the next display:

MAIN DISHES (DINNER)
VEGETARIAN NON-VEGETARIAN

Since fish is non-vegetarian:

GET NON-VEGETARIAN (return)

The computer will then show you this display:

MAIN DISHES (DINNER, NON-VEGETARIAN)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Since you are looking for a menu that contains fish, your next response would be: GET SEAFOOD [return]

The computer still has menus in the recipe database, and it will print out:

MAIN DISHES (DINNER, NON-VEGETARIAN, SEAFOOD)
SEAFOOD FISH TUNA

Since the problem had asked for a recipe containing fish, your next choice could be:

GET FISH (return)

and the computer will give a listing of the recipes it has in database listed under fish:

BROILED FISH
CODFISH BALL
DEEP FRIED FISH
FLOUNDER PROVENCALE
FRIED FISH
HADDOCK SHRIMP BAKE
HALIBUT ROYALE
HERB BAKED FISH
OVEN FRIED FISH
SALMON LOAF
SALMON STEAK
SCALLOPED FISH
STEAMED FISH
STUFFED FISH
STUFFED WHITEFISH
FISH IN A BASKET
GRILLED HALIBUT
HICKORY FISH BAKED
FILLETS ELEGANTE

The final menu is made choosing a name from the database listed. For example:

GET DEEP FRIED FISH (return)

This is called a final menu and will consist of the name of exactly one recipe.

You are permitted to make a sequence of selection from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose you had selected item 3, Deep Fried Fish, from the recipe database list of fish, above. This would cause the menu consisting only of deep fried fish to appear, i.e.:

DEEP FRIED FISH (return)

Now, suppose that you want to see the recipe for Flounder Provencale. This was item 4 on the previous menu. You can directly request that the recipe for the "Next Menu Item" be shown. If the next item is the name of a single recipe (i.e., Flounder

Provencale), you can move directly to that recipe by entering:

NEXT (return)

Then the recipe for Flounder Provencale would appear. But if the next item represented several recipes (e.g., Fish) then a new menu would appear instead. Note that if you request the "Next Menu Item" but are already at the bottom of the menu, then an error will result.

Say that you have run through the database, and you are now back at the menu that looks like this:

MAIN DISHES (DINNER, NON-VEGETARIAN)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Because you have tried this earlier, you remember that when you choose seafood, you then will have a choice of seafood, fish, or tuna. Last time "fish" was chosen, and now you are interested in seeing what is listed under "tuna". To do this, enter:

GET SEAFOOD, TUNA (return)

and the menu asking you to choose tuna will be skipped and you will go directly to what the database has under "tuna".

Suppose you make a mistake and choose the wrong item from a menu. For example, maybe you chose "desserts" when you really wanted "Main Dishes". You can always "start over" completely by going back directly to the top menu. To do this at any time, just enter:

NEXT (return)

Just remember that the "TOP" option always lets you start over from the first (top) menu.

Example of command usage follows:

To choose the next or previous item in a menu (as described above) enter:

NEXT (return)
or
PRIOR (return)

To go to the top menu (say, choice 3, Deep Fried Fish) from the current menu, enter:

GET DEEP FRIED FISH,RECIPE CHOICE (return)

Note: Only one choice can be made. Otherwise an error should result.

Also note that choices are entered BY SPELLING OUT THE CHOICE, NOT by entering the item number.

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

How to Correct Some Mistakes

If you make a mistake and ask for something that is not on the menu, the computer will tell you that it can't figure out what you want and will ask you to re-enter your choice. For example, if you entered GET CLD, the computer would not be able to find what you wanted so you would have to re-enter.

Your main objective in finding an item you want is to keep selecting until you get down to a menu with just ONE item. Suppose you make a mistake and choose the wrong item from a menu. For example, maybe you chose "Main Dishes" when you really wanted "Desserts". You can "go back" in three ways. These are each slightly different so it is important to carefully understand the three ways of correcting mistakes.

First, you can always "start over" completely by going back directly to the top menu. To do this at any time, just enter:

TOP (return)

Second, you can always go back to the menu you just saw (i.e., the last menu you saw before the one that is now on the screen), by using the P (for Previous Menu) option:

BACK UP (return)

If you keep using the Back Up option several times in a row, you will see that you just keep going back and forth between the same two menus (because one was always the "last" menu you saw before the one that is on the screen). To see this, try entering BACK UP several times.

You can see that the first time you enter Back Up you went back to the top menu. The next time you went back to the "Breakfast, Lunch, Dinner" menu. So the P option lets you go both "forward" (i.e., to the next level down--the "Breakfast, Lunch, Dinner" menu is one level down below the top menu) and "backward" (i.e., go to the top menu from the "Breakfast, Lunch, Dinner" menu).

The last way to go back to a menu and correct a choice is to use the Up option (for Higher level). This always takes you back to the menu one step above the menu on the screen. For example, if the "Dessert-type" menu list was on the screen (e.g., Drop Cookies) then entering:

UP (return)

will get you back to the top menu (Desserts, Cookies; Bar, Drop, Filled, Shaped). Then if you entered UP again, this would get you to the top menu (Desserts; Cakes, Candies, Cookies, Custards, Frozen, Fruit, Pies, Puddings) since this is the menu above the Desserts, Cookies; Bar, Drop, Filled, Shaped.

If you get to the top menu, UP will not have any further effect since the top menu is as far as you can go.

Notice that UP keeps moving you up toward the top menu. But BACK UP would just keep moving you between two menu levels. So the BACK UP option is good for moving back one menu (in either direction) while the UP option is best for moving one or more levels back toward the top menu.

Once you get to the menu you want, you can just select a new choice from it.

How to Get the Message "Good Choice"

Normally, you should keep making menu choices until you narrow the menus to one recipe item, the recipe you want to choose for your menu plan. Use the "Recipe Choice" option by typing:

GET DEEP FRIED FISH,RECIPE CHOICE (return)

After you receive the message "GOOD CHOICE" you can continue in several ways. You could enter UP, for example, to start over by going back to the top menu. You could also type BACKUP or TOP to return to the previous menu you were looking at.

Note: Only one choice can be made. Otherwise an error should result.

Also note that choices are entered BY SPELLING OUT THE CHOICE, NOT by entering the item number.

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

Now you would write the name of the recipe on your answer worksheet. If you have completed all the segments of the problem, indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems then enter:

DONE (return)

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 10. Single, Large, No Backup

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning using food recipes. So the information you will trying to get will be names of different food dishes. For example, you might be asked to plan a dinner which includes fish, a vegetable, and some form of potato.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipe choices to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the recipe database to solve the problem. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful. (But don't worry about forgetting the details of the problem--you can see the problem again whenever you wish by entering the command "PROBLEM" [return].)

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

For example, the problem is to plan a dinner menu that includes fish.

Following is the list of commands you will be using to solve all the problems.

GET ITEM1.....SELECT ITEM1 ON MENU
RECIPE CHOICE....PRESENTS THE MESSAGE "GOOD CHOICE"
TOP.....GO TO TOP MENU
NEXT.....SELECT NEXT MENU ITEM AND DISPLAY
PRIOR.....SELECT PRIOR MENU ITEM AND DISPLAY

Just to get started, when you see ENTER COMMAND: at the bottom of your screen, you will use the command:

GET TOP (return)

The following menu will then be displayed on your monitor:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

You can then choose the one you think will be most helpful in solving the problem. In this case, you would enter:

GET MAIN DISHES (return)

The computer will then display the following sub-menu on its screen:

MAIN DISHES
BREAKFAST LUNCH DINNER

Your choice from this menu would be:

GET DINNER (return)

because you are searching for a dinner menu. The computer will now give you the next display:

MAIN DISHES (DINNER)
VEGETARIAN NON-VEGETARIAN

Since fish is non-vegetarian:

GET NON-VEGETARIAN (return)

The computer will then show you this display:

MAIN DISHES (DINNER, NON-VEGETARIAN)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Since you are looking for a menu that contains fish, your next response would be:

GET SEAFOOD (return)

The computer still has sub-menus in the recipe database, and it will print out:

MAIN DISHES (DINNER, NON-VEGETARIAN, SEAFOOD)
SEAFOOD FISH TUNA

Since the problem had asked for a recipe containing fish, your next choice could be:

GET FISH (return)

and the computer will give you a listing of the recipes it has in the database listed under fish:

BROILED FISH
CODFISH BALL
DEEP FRIED FISH
FLOUNDER PROVENCALE
FRIED FISH
HADDOCK SHRIMP BAKE
HALIBUT ROYALE
HERB BAKED FISH
OVEN FRIED FISH
SALMON LOAF
SALMON STEAK
SCALLOPED FISH
STEAMED FISH
STUFFED FLOUNDER
STUFFED WHITEFISH
FISH IN A BASKET
GRILLED HALIBUT
HICKORY FISH BAKED
FILLETS ELEGANTE

The final menu is made by choosing an item from the above list, for example:

GET DEEP FRIED FISH (return)

This is called a final menu and will consist of the name of exactly one recipe.

You are permitted to make a sequence of selections from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose you had selected item 3, Deep Fried Fish, from the recipe database list of fish, above. This would cause the menu consisting only of Deep Fried Fish to appear, i.e.:

DEEP FRIED FISH (return)

Now, suppose that you want to see the recipe for Flounder Provencale. This was item 4 on the previous menu. You can directly request that the recipe for the "Next Menu Item" be shown. If the next item is the name of a single recipe (i.e., Flounder Provencale), you can move directly to that recipe by entering:

NEXT (return)

Then the recipe for Flounder Provencale would appear. But if the next item represented several recipes (e.g., Fish), then a new menu would appear instead. Note that if you request the "Next Menu Item" but are already at the bottom of the menu, then an error will result.

Say that you have run through the database, and you are now back at the menu that looks like this:

MAIN DISHES (DINNER, NON-VEGETARIAN)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Because you have tried this earlier, you remember that when you choose seafood, you will then have a choice of seafood, fish, or tuna. Last time "fish" was chosen, and now you are interested in seeing what is listed under "tuna". To do this, enter:

GET SEAFOOD, TUNA (return)

and the menu asking you to choose tuna will be skipped and you will go directly to what the database has under "tuna"

Suppose you make a mistake and choose the wrong item from a menu. For example, maybe you chose "Desserts" when you really wanted "Main Dishes". You can always "start over" completely by going back directly to the top menu. To do this at any time, just enter:

NEXT (return)

Just remember that the "TOP" option always lets you start over from the first (top) menu.

Example of command usage follow:

To choose the next of previous item in a menu (as described above) enter:

NEXT (return)
or
PRIOR (return)

To go to the top menu (i.e., the level 1 menu), enter:

TOP (return)

To select a choice (say, Deep Fried Fish) from the current menu, enter:

GET DEEP FRIED FISH,RECIPE CHOICE (return)

Note: Only one choice can be made. Otherwise an error should result.

Also note that choices are entered BY SPELLING OUT THE CHOICE.

If the choice is appropriate, the computer will give you the message:

"GOOD CHOICE!"

Now you would write the name of the recipe on your answer worksheet. If you have completed all the segments of the problem, indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems then enter:

DONE (return)

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 11. Multiple, Large, Backup

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning using food recipes. So the information you will trying to get will be names of different food dishes. For example, you might be asked to plan a dinner which includes fish, a vegetable, and some form of potato.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipe choices to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the recipe database to solve the problem. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful. (But don't worry about forgetting the details of the problem--you can see the problem again whenever you wish by entering the command "PROBLEM" [return].)

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

For example, the problem is to plan a dinner menu that includes fish.

Following is the list of commands you will be using to solve all the problems.

GET.....SELECT ITEM1 ON MENU
JUMP ITEM1, ITEM2....SELECT ITEM1 ON MENU; ITEM2 ON NEXT MENU
RECIPE CHOICE.....PRESENTS THE MESSAGE "GOOD CHOICE"
BACKUP.....GO TO PREVIOUS MENU
UP.....GO TO NEXT HIGHER MENU
NEXT.....SELECT NEXT MENU ITEM
PRIOR.....SELECT PRIOR MENU ITEM
TOP.....GO TO TOP MENU

Just to get started, when you see: ENTER COMMAND: at the bottom of your screen, you will use the command:

TOP (return)

The following menu will then be display on your monitor:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

You can then choose the one you think will be most helpful in solving the problem. In this case, you would enter:

GET MAIN DISHES (return)

The computer will then display the following sub-menu on its screen:

MAIN DISHES (DINNER)
VEGETARIAN NON-VEGETARIAN

Since fish is non-vegetarian:

GET NON-VEGETARIAN (return)

The computer will then show you this display:

MAIN DISHES (DINNER, NON-VEGETARIAN)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Since you are looking for a menu that contains fish, your next response would be:

GET SEAFOOD (return)

The computer still has sub-menus in the recipe database, and it will print out:

MAIN DISHES (DINNER, NON-VEGETARIAN, SEAFOOD)
SEAFOOD FISH TUNA

Since the problem had asked for a recipe containing fish, your next choice could be:

GET FISH (return)

and the computer will give you a listing of the recipes it has in the database listed under Fish:

BROILED FISH
CODFISH BALL
DEEP FRIED FISH
FLOUNDER PROVENCALE
FRIED FISH
HADDOCK SHRIMP BAKE
HALIBUT ROYALE
HERB BAKED FISH
OVEN FRIED FISH
SALMON LOAF
SALMON STEAK
SCALLOPED FISH
STEAMED FISH
STUFFED FLOUNDER
STUFFED WHITEFISH
FISH IN A BASKET
GRILLED HALIBUT
HICKORY FISH BAKED
FILLETS ELEGANTE

The final menu is made by choosing a name from the database list, for example:

GET DEEP FRIED FISH (return)

This is called a final menu and will consist of the name of exactly one recipe.

You are permitted to make a sequence of selections from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose you had selected item 3, Deep Fried Fish, from the recipe database of fish, above. This would cause the menu consisting only of Deep Fried Fish to appear, i.e.: DEEP FRIED FISH

In order to complete the solution to the part of the problem you MUST receive the message "GOOD CHOICE" from the computer.

How to Get the Message "Good Choice"

Normally, you should keep making menu choices until you narrow the menus to one recipe item -- the recipe you want to choose for your menu plan. Use the "Recipe Choice" option by typing:

GET FRIED FISH,RECIPE CHOICE (return)

After you receive the message "Good Choice" you can continue in several ways. You could enter TOP, for example, to start over by going back to the top menu. Also type BACK UP for the previous

menu; or you may enter UP to return to the next higher menu.

The message "GOOD CHOICE" is an indication to you that you have solved part of the problem and can continue working on next part of the problem or start the new problem.

How to Change Your Menu Choice

Now, suppose that you want to see the recipe for Flounder Provencale. This was item 4 on the previous menu. You can directly request that the recipe for the "Next Menu Item" be shown. If the next item is the name of a single recipe (i.e., Flounder Provencale), you can move directly to that recipe by entering:

NEXT (return)

Then the recipe for Flounder Provencale would appear. But if the next item represented several recipes (e.g., Fish), then a new menu would appear instead. Note that if you request the "Next Menu Item" but are already at the bottom of the menu, then an error will result.

Similarly, if you want to choose COD FISH BALL, you can use the command:

PRIOR (return)

How to Correct Some Mistakes

If you make a mistake and ask for something that is not on the menu, the computer will tell you that it can't figure out what you want and will ask you to re-enter your choice. For example, if you entered OLD, the computer would not be able to find what you wanted so you would have to re-enter.

Your main objective in finding an item you want is to keep selecting until you get down to a menu with just ONE item. Suppose you make a mistake and choose the wrong item from a menu. For example, maybe you chose "Main Dishes" when you really wanted "Desserts". You can "go back" in three ways. These are each slightly different so it is important to carefully understand the three ways of correcting mistakes.

First, you can always "start over" completely by going back directly to the top menu. To do this at any time, just enter:

TOP (return)

Second, you can always go back to the menu you just saw (i.e., the last menu you saw before the one that is now on the screen) by using the BACKUP (to previous menu) option:

BACK UP (return)

If you keep using the BACK UP option several times in a row, you will see that you just keep going back and forth between the same two menus (because one was always the "last" menu you saw before the one that is on the screen). To see this, try entering BACK UP several times.

For example, if you selected MAIN DISHES from TOP menu, the computer will display:

MAIN DISHES
BREAKFAST LUNCH DINNER

At this stage, if you use the BACKUP command, you will see the TOP menu, that is:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

If you press the BACKUP command again, you will see MAIN DISHES:

MAIN DISHES
BREAKFAST LUNCH DINNER

You can see that the first time you enter BACK UP you went back to the top menu. The next time you went back to the "Breakfast, Lunch, Dinner" menu. So the BACK UP option lets you go both "forward" (i.e., to the next level down--the "Breakfast, Lunch, Dinner" menu is one level down below the top menu) and "backward" (i.e., go to the top menu from the "Breakfast, Lunch, Dinner" menu).

The last way to go back to a menu and correct a choice is to use the H option (for Higher level). This always takes you back to the menu one step above the menu on the screen. For example, if the "Dessert-type" menu list was on the screen (e.g., Drop Cookies) then entering:

UP (return)

will get you back to the top menu (Desserts, Cookies; Bar, Drop, Filled, Shaped). Then if you entered UP again, this would get you to the top menu (Desserts; Cakes, Candies, Cookies, Custards, Frozen, Fruit, Pies, Puddings) since this is the menu above the (Desserts, Cookies; Bar, Drop, Filled, Shaped).

If you get to the top menu, UP will not have any further effect since the top menu is as far as you can go.

Notice that UP keeps moving you up toward the top menu. But BACK UP would just keep moving you between two menu levels. So the BACK UP option is good for moving back one menu (in either direction) while the UP option is best for moving one or more levels back toward the top menu.

Once you get to the menu you want, you can just select a new choice from it.

Combining Options

Options such as Recipe Choice and Back Up can be combined to make the GET command work more effectively. For example, suppose you just got the message "GOOD CHOICE!", but you decide against that choice, and want to choose seafood option. One way to do this is to type BACK UP to backup to the last menu you see, and then UP to go up one more menu, to the next higher menu.

An easier way to do this is to combine the options and just type:

BACK UP,UP,NEXT,RECIPE CHOICE (return)

You should be able to see how these options have been combined. The only new thing is the "Next" sign. This means "Next Item in Menu". You could have typed:

BACK UP, SEAFOOD (return)

but since you know that you just wanted to check the next item in the menu, the "Next" was used instead of the full name. This is helpful especially if you cannot remember what the exact name of the next menu item was. You can also use the "Prior" symbol to mean the "Previous Item on the Menu". For example, if you had chosen CODFISH BALLS

BACK UP,UP,PRIOR,CODFISH BALLS,RECIPE CHOICE (return)

would give you the message "GOOD CHOICE!"

Here is another example of combining options. Suppose you had a menu that contained several items. Normally, you would first type GET CODFISH BALLS to choose this item and form a new menu with just one item, CODFISH BALLS. Then you would use the GET RECIPE CHOICE option to get the message "Good Choice!" However, you could combine these steps by typing the combination shown here:

GET CODFISH BALLS,RECIPE CHOICE (return)

You can see that this first selected CODFISH BALLS, RECIPE CHOICE and then displayed the message "GOOD CHOICE!"

Remember, you can combine options in any other way that you think will help you get to the information you want more easily.

Jump Item1, Item2

This command can be used to select items from more than one menu. For example, you want a dinner menu to solve a problem. You

will first use the command TOP to get to the top menu. Then, out of that menu you will select MAIN DISHES and after that you will select the DINNER option. The command JUMP ITEM1, ITEM2... allows you to make the selection from different levels simultaneously. For example, after you get the top menu, you can use this command by typing:

GET MAIN DISHES,DINNER (return)

This will give you the dinner menu. This will only work if you remember what the next menu is going to be. Otherwise the computer will give you an error message.

Now you would write the name of the recipe on your answer worksheet and indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems then enter:

DONE (return)

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

Instructions

Interface 12. Multiple, Large, No Backup

This is an experiment to study how to make computers easier to use.

In this experiment you will be asked to solve a series of four separate problems. For each problem you will have to get information out of the computer. The problems are about menu planning using food recipes. So the information you will trying to get will be names of different food dishes. For example, you might be asked to plan a dinner which includes fish, a vegetable, and some form of potato.

A worksheet is provided for each problem. As you get food recipes out of the computer, you might want to jot down notes. After you feel satisfied that you have the best set of recipe choices to solve a problem, make sure you write down the recipe names you have chosen on the worksheet so we will have a record of your final menu plan for each problem.

Now you will be shown how to use the recipe database to solve the problem. Try to follow the examples closely and be sure to ask the experimenter if you have any questions.

First, a problem will be automatically presented to you on the display. You do not have to do anything except wait until the problem shows up. When the problem appears, read it carefully. Feel free to jot down any notes about the problem that you feel might be helpful. (But don't worry about forgetting the details of the problem--you can see the problem again whenever you wish by entering the command "PROBLEM" [return].)

After you have studied the problem, press any key to begin working on it. Shortly after you press a key to begin work, the display screen will change. At the bottom left corner of the display are the words "ENTER COMMAND:". This is where commands TO the computer are entered. The computer will work on each command you enter and will try to carry out your instructions.

Study the information that comes back from the computer. Then, when you are ready, enter your next command. You will keep giving commands and getting information back until you have solved the problem to your own satisfaction. Then, there is a special command to tell the computer that you are done with the current problem and ready to go on to the next problem.

For example, the problem is to plan a dinner menu that includes fish.

Following is the list of commands you will be using to solve all the problems.

```
GET.................SELECT ITEM1 ON MENU
JUMP ITEM1, ITEM2....SELECT ITEM1 ON MENU; ITEM2 ON MENU2, ETC.
RECIPE CHOICE........PRESENTS THE MESSAGE "GOOD CHOICE"
NEXT.................SELECT NEXT MENU ITEM
PRIOR................SELECT PRIOR MENU ITEM
TOP..................GO TO TOP MENU
```

Just to get started, when you see: ENTER COMMAND: at the bottom of your screen, you will use the command:

TOP (return)

The following menu will then be display on you monitor:

APPETIZERS SOUPS SALADS MAIN DISHES
DESSERTS SIDE DISHES BEVERAGES

You can then choose the one you think will be most helpful in solving the problem. In this case, you would enter:

GET MAIN DISHES (return)

The computer will then display the following sub-menu on its screen:

MAIN DISHES (DINNER)
VEGETARIAN NON-VEGETARIAN

Since fish is non-vegetarian:

GET NON-VEGETARIAN (return)

The computer will then show you this display:

MAIN DISHES (DINNER, NON-VEGETARIAN)
GROUND MEAT BEEF VEAL PORK
POULTRY LAMB SEAFOOD

Since you are looking for a menu that contains fish, your next response would be:

GET SEAFOOD (return)

The computer still has sub-menus in the recipe database, and it will print out:

MAIN DISHES (DINNER, NON-VEGETARIAN, SEAFOOD)
SEAFOOD FISH TUNA

Since the problem had asked for a recipe containing fish, your next choice could be:

GET FISH (return)

and the computer will give you a listing of the recipes it has in the database listed under fish:

BROILED FISH
CODFISH BALL
DEEP FRIED FISH
FLOUNDER PROVENCALE
FRIED FISH
HADDOCK SHRIMP BAKE
HALIBUT ROYALE
HERB BAKED FISH
OVEN FRIED FISH
SALMON LOAF
SALMON STEAK
SCALLOPED FISH
STEAMED FISH
STUFFED FLOUNDER
STUFFED WHITEFISH
FISH IN A BASKET
GRILLED HALIBUT
HICKORY FISH BAKED
FILLETS ELEGANTE

The final menu is made by choosing a name from the database list, for example:

GET DEEP FRIED FISH (return)

This is called a final menu and will consist of the name of exactly one recipe.

You are permitted to make a sequence of selections from a menu without actually having to return to the menu display each time a new selection is desired. For example, suppose you had selected item 3, Deep Fried Fish, from the recipe database of fish, above. This would cause the menu consisting only of Deep Fried Fish to appear, i.e.: DEEP FRIED FISH

Now, suppose that you want to see the recipe for Flounder Provencale. This was item 4 on the previous menu. You can directly request that the recipe for the "Next Menu Item" be shown. If the next item is the name of a single recipe (i.e., Flounder Provencale), you can move directly to that recipe by entering:

NEXT (return)

Then the recipe for Flounder Provencale would appear. But if the next item represented several recipes (e.g., Fish), then a new menu would appear instead. Note that if you request the "Next Menu Item" but are already at the bottom of the menu, then an error will result.

Similarly, if you want to choose COD FISH BALL, you can use the command:

PRIOR (return)

How to Get the Message "Good Choice"

Normally, you should keep making menu choices until you narrow the menus to one recipe item, the recipe you want to choose for your menu plan. Use the "RECIPE CHOICE" option by typing:

GET DEEP FRIED FI: RECIPE CHOICE (return)

If the choice is appropriate, the computer will give you the message "Good Choice!"

How to Correct Some Mistakes

If you make a mistake and ask for something that is not on the menu, the computer will tell you that it can't figure out what you want and will ask you to re-enter your choice. For example, if you entered GET CODFISH BALL, the computer would not be able to find what you wanted so you would have to re-enter.

Your main objective in finding an item you want is to keep selecting until you get down to a menu with just ONE item. You can always "start over" completely by going back directly to the Top menu. To do this at any time, just enter:

TOP (return)

Jump Item1, Item2

This command can be used to select items from more than one menu. For example, you want a dinner menu to solve a problem. You will first use the command TOP to get to the top menu. Then out of that menu you will select MAIN DISHES and after that you will select the DINNER option. The command ITEM1, ITEM2... allows you to make the selection from different levels simultaneously. For example, after you get the top menu, you can use this command by typing:

GET MAIN DISHES,DINNER (return)

This will give you the dinner menu. This will only work if you remember what the next menu is going to be. Otherwise the computer will give you an error message.

Now you would write the name of the recipe on your answer worksheet and indicate you are ready for the next problem by entering:

NEW PROBLEM (return)

This would complete the problem.

When you have completed all the problems then enter:

DONE (return)

You should carefully review these instructions and be sure to ask the experimenter to explain anything about which you are unsure. Of course, it will take some practice to get used to all the commands in this computer system. Try to solve each problem in the way that is easiest for you. Using the system will get easier as you go through the problems because you will become more familiar with the commands.

When you have finished these instructions and are ready to begin, let the experimenter know by entering the command:

READY (followed by "return" key)

REFERENCES

Andriole, S.J. "The Design of Microcomputer-Based Personal Decision-Aiding Systems" IEEE Transactions on Systems, Man, and Cybernetics, 1982, SMC-12, 463-469.

Ambardar, Anita Kak. Individual Difference Effects in Human-Computer Interaction, Technical Report, U.S. Army Contract MDA 903-82-c-0157, November, 1982.

Ambardar, Anita Kak. Individual Difference Effects in Human-Computer Interaction, Technical Report, Part 1, U.S. Army Contract MDA 903-82-c-0157, 1983.

Ambardar, Anita Kak. Individual Difference Effects in Human-Computer Interaction, Technical Report, Part 2, U.S. Army Contract MDA 903-82-c-0157, 1983.

Bariff, M.L. and Lusk, E.J. "Cognitive and Personality Tests for the Design of Management Information Systems" Management Science, 1977, 23, 820-829.

Benbasat, Izak and Dexter, Albert S. "Value and Events Approaches to Accounting: An Experimental Evaluation" The Accounting Review, 54, 735-749.

Benbasat, Izak and Dexter, Albert S. "Individual Differences in the Use of Decision Support Aids. Journal of Accounting Research, 1982, 20, 1-11.

Benbasat, Izak and Taylor, Richard N. "Behavioral Aspects of Information Processing for the Design of Management Information Systems" IEEE Transactions on Systems, Man, and Cybernetics, July-August 1982, Volume SMC-12, No 4, 439-450.

Barnard, P.J., Hammond, N.V., Morton, J., Long, J.B. and Clark, I.A. "Consistency and Compatibility in Human-Computer Dialogue" International Journal of Man-Machine Studies, 1981, 14, in press.

Bieri, J. "Cognitive Complexity and Personality Development" In O.J. Harvey (ed.), Experience, Structure and Adaptability, New York: Springer, 1966.

Bourne, L.E. Jr., Ekstrand, B.R., and Dominowski, R.L. "The Psychology of Thinking" New Jersey: Prentice-Hall, Inc., 1971.

Broadbent, D.E. "Stimulus Set and Response Set: Two Kinds of Selective Attention" In D.I. Mostofsky (Ed.), Attention: Contemporary Theories and Analysis, New York: Appleton-Century-Croft, 1970.

Broverman, D.M., "Dimensions of Cognitive Style" Journal of Personality, 1960b, 28, 167-185.

Broverman, D.M., "Generality and Behavior Correlates of Cognitive Styles" Journal of Consulting Psychology, 1964, 28, 487-500.

Coop, A.H., and Sigel, I.E., "Cognitive Style: Implications for Learning and Instruction" Psychology in the Schools, 1971, 2, 152-161.

Davis, J.K., "Conditional Concept Learning and Cognitive Style" Perceptual and Motor Skills, 1975, 40, 859-862.

Davis, J.K. and Frank, B.M., "Learning and Memory of Field Independent and Dependent Individuals" Journal of Research in Personality, 1979, 13, 469-479.

Davis, J.K. and Haueisen, W.C., "Field Independence and Hypothesis Testing" Perceptual and Motor Skills, 1976, 43, 736-769.

Davis, J.K. and Klausmeier, H.J., "Cognitive Style and Concept Identification as a Function of Complexity and Training Procedures" Journal of Educational Psychology, 1970, 61, 423-430.

Dickstein, L.S., "Field Independence in Concept Attainment" Perceptual and Motor Skills, 1968, 27, 635-642.

Eagle, M., Fitsgibbons, D., and Goldberger, L., "Field Dependence and Memory for Relevant and Irrelevant Incidental Stimuli" Perceptual and Motor Skills, 1966, 23, 1035-1038.

Eimas, P.D., "Effects of Memory Aids on Hypothesis Behavior and Focusing in Young Children and Adults" Journal of Experimental Child Psychology, December 1970, 10(3), 319-336.

Flemming, M.L., "Message Design: The Temporal Dimension of Message Structure" Bloomington, IN: U.S. Office of Education, Contract No. USDE 7-24-0210-282, Audio-Visual Center, Indiana University, March, 1968.

Gates, D.W., "Verbal Conditioning, Transfer and Operant Level 'Speech Style' as Functions of Cognitive Style" Dissertation Abstracts International, 1971, 32(6-B), 3634.

Gardner, R.W., "Cognitive Styles in Categorizing Behavior" Journal of Personality, 1953, 22, 214-233.

Gardner, R.W., Holzman, P.S., Klein, G.S., Linton, H.B., and Spence, D.P., "Cognitive Control: A Study of Individual Consistencies in Cognitive Behavior" Psychological Issues, 1959, 1(4 Whole No 4).

Gardner, R.W., Jackson, D.N., and Messick, S.J., "Personality Organization in Cognitive Controls and Intellectual Abilities" Psychological Issues, 1960, 2, 1-148.

Gardner, R.W., Lohrenz, L.J., and Schoen, R.A., "Cognitive Control of Differentiation in the Perception of Persons and Objects" Perceptual and Motor Skills, 1968, 26, 311-330.

Gardner, R.W., and Long, R.I., "Cognitive Controls as Determinants of Learning and Remembering" Psychologia, 1960a, 3, 155-171.

Gardner, R.W., and Schoen, R.A., "Differentiation and Abstraction in Concept Formation" Psychological Monographs, 1962, 76(41, Whole No 560).

Gollin, E.S., "Forming Impressions of Personality" Journal of Personality, 1954, 23, 65-76.

Goodenough, D.R., "The Role of Individual Differences in Field Dependence as a Factor in Learning and Memory" Psychological Bulletin, 1976, 83, 675-694.

Grippen, P.C., and Ohnmact, F.W., "Relationship of Field Independence and Dogmatism with an Hierarchically-Arranged Concept-Learning Task" Perceptual and Motor Skills, 1972, 34, 983-986.

Henderson, John C. and Nutt, Paul C., "The Influecne of Decision Style on Decision Making Behavior" Management Science, 1980, 26, 371-386.

Huang, M.S., "Category Width and Individual Differences in Information Processing Strategies" Journal of Psychology, 1981, 108, 73-79.

Hunt, E. "Intelligence as an Information-Processing Concept" British Journal of Psychology, November 1980, Volume 71, Pt 4, 449-474.

Innocent, P.R., "Toward Self-Adaptive Interface Systems" International Journal of Man-Machine Studies, 1982, 16, 287-299.

Kagan, J., Moss, H.A., and Sigel, I.E., "Psychological Significance of Styles of Conceptualization" Monogr. Soc. Res. Child Dev., 1963, 28, No 86.

Kagan, J., Rosman, B.L., Day, D., Albert, J., and Phillips, W., "Information Processing in the Child: Significance of Analytic and Reflective Attitudes" Psychological Monographs: General and Applied, 1964, 78(1, Whole No 578).

Klein, G.S., "Perception, Motives, and Personality" New York: Knopf, 1970.

Konstadt, N., and Forman, E., "Field Dependence and External Directedness" Journal of Personality and Social Psychology, 1965, 1, 490-493.

Lusk, Edward J., "A Test of Differential Performance Peaking for a Disembedding Task" Journal of Accounting Research, 17, 286-294.

Massari, D.J., and Mansfield, R.S., "Field Dependence and Outer Directedness in the Problem Solving of Retardates and Normal Children" Child Development, 1973, 44(2), 346-358.

Martin, J., "The Design on Man-Computer Dialogues" Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

Messick, S., and Associates, "Individuality in Learning" San Francisco: Jossey-Bass, 1976.

Messmer, O., "Zur Psychologie des Lesens bei Kindern und Erwachesenen" Arch. ges. Psychol., 1903, 2, 190-298.

Meumann, E., "Vortrag z. Einf." in der Experimentalischer Padagogik, 1907.

Miller, R.B., "Archetypes in Man-Computer Problem Solving" Ergonomics, 1969, 12, 559-581.

Nebelkopf, E.B., and Dreyer, A.S., "Continuous-Discontinuous Concept Attainment as a Function of Individual Differences in Cognitive Style" Perceptual and Motor Skills, 1973, 36, 655-662.

O'Connor, K.P., and Blowers, G.H., "Cognitive Style, Set, and Sorting Strategy" British Journal of Psychology, 1980, 71, 17-22.

Ohnmact, F.W., "Effects of Field Independence and Dogmatism on Reversal and Non-Reversal Shifts in Concept Formation" Perceptual and Motor Skills, 1966, 22, 491-497.

Rorschach, H., "Psychodiagnostik: Methodik und Ergebnisse eines Warhneh-Mungsdiagnostischen Experiments" Bircher, Bern., 1921.

Rosecrans, C.J., "The Relationship Between Perceptual Performance and Three Types of Learning Tasks" (Doctoral Dissertation, University of Tennessee) Ann Arbor, Michigan: University Microfilms, 1955, No 13, 063.

Sackman, H., "Improving the Human Factors Aspect of Database Interactions" ACM Transaction Database Systems, December 1978, Volume 3, No 4.

Schroder, H.M., "Conceptual Complexity and Personality Organization" In H.M. Schroder and P. Suedfeld (Eds.), Personality Theory and Information Processing, New York: Ronald, 1971.

Shapson, S.M., "Hypothesis Testing and Cognitive Style in Children" Journal of Educational Psychology, August 1977, 69, 4, 452-463.

Suedfeld, P., "Information Processing as a Personality Model" In H.M. Schroder and P. Sudefeld (Eds.), Personality Theory and Information Processing, New York: Ronald, 1971.

Thomas, J.C., "A Design Interpretation Analysis of Natural English with Application to Man-Computer Interaction

Van Veen, W.J., et. al., "Temporal Characteristics of the Contingent Negative Variation: Relationships with Anxiety, Perceptual Mode, Sex and Stress" Biological Psychiatry, 1973, 7(2), 101-111.

Vernon, P.E., "Multivariate Approaches to the Study of Cognitive Styles" In J.E. Royce (Ed.), Multivariate Analysis and Psychology, New York: Academic Press, 1973, 125-148.

Witkin, H.A., and Dyk, R.B., "Psychological Differentiation: Studies of Development" Wiley, New York, 1962.

Witkin, H.A., Oltman, P.K., Raskin, E., and Karp, S.A., "Manual for Embedded Figures Test, Children's Embedded Figures Test, and Group Embedded Figures Test" Palo Alto, California: Consulting Psychologists Press, Inc., 1971.

Witken, Vernum A., et.al., "Field-Dependent and Field-Independent Cognitive Styles and Their Educational Implications" Research Bulletin, Princeton, New Jersey: Educational Testing Services, 1975, 75-24.

Witkin, H.A., Dyk, R., Faterson, H., Goodenough, D., and Karp, "Psychological Differentiation" Lawrence Erlbaum Associates: Potomac, MD., 1974.

Zajonc, R.B., "Cognitive Theories in Social Psychology" In G. Lindzey and E. Aronson (Eds.), The Handbook of Social Psychoalogy, Reading, Mass: Addison-Wesley, 1968, Volume 1.

Zawel, D., "The Effects of Varying Degrees of Field Dependence on Discrimination Learning" Dissertation Abstracts International, 1969, 30(3-B), 1351.

Bechhofer, R.E., "A Single-Sample Multiple Decision Procedure for Ranking Means of Normal Populations with Known Variance" Ann. Math. Statist., 1954, 25, 16-39.

Gupta, S.S., "On a Decision Rule for a Problem in Ranking Means" Mimeo Series No. 150, Inst. of Statist., University of North Carolina, Chapel Hill, 1956.

Gupta, S.S., "On Some Multiple Decision (Selection and Ranking) Rules" Technometrics, 1965.

Hsu, J.C., "Simultaneous Confidence Intervals for all Distances from the Best" Ann. Statist., 1981, 9, 1026-1034.

Hsu, J.C., "Ranking and Selection and Multiple Comparisons with the Best" Design of Experiments: Ranking and Selection, T.J. Santner and A. Tamhane (Eds.), Marcel Dekker, New York, 1984a.

Hsu, J., "Constrained Simultaneous Confidence Intervals for Multiple Comparisons with the Best" Ann. Statist., 1984b, 12, 1136-11444.